MENTAL MODELS

The Corner Shelf

I've been collecting mental models for years. Most of them I read once, nodded at, and never thought about again. These twenty-four are what's left. They're the ones I can still hear in a meeting that's going badly, before anyone has worked out why.

I've grouped them by what they do rather than where they came from, because that's how they turn up. Nobody reaches for a category in the middle of an argument. You reach for the one question that would have saved the last three weeks.

A model is a compression. It throws away most of a situation so one part of it becomes visible, the way a map throws away everything that is not the road. That is what makes them useful and it is also why each one is wrong somewhere. A map that showed everything would be the size of the place and no use to anybody.

Two ways to use the list. If a model here is new to you, read that one entry and ignore the rest. And when a decision is stuck, go down the index and ask which of these the problem is shaped like. That second one is what the index is for, and it works better than trying to remember the right model at the moment you need it.

None of it is a checklist. A model is a way of standing somewhere else for a minute and seeing what the problem looks like from there.

The where it breaks notes are the part I'd keep if I had to cut everything else. Trusting a model past its edge does more damage than never having picked it up.

Contents

I Framing the problem

Before anything else, the question is usually wrong. These three change it.

```
how do we win?      →  vague
what makes us lose? →  a list
```

01 Inversion

Don't ask how to succeed. Ask what would guarantee failure, then refuse to do it.

Jacobi said it in three words. *Man muss immer umkehren.* Invert, always invert. It helps because most planning questions are open at one end. Ask how do we grow adoption and the answer space has no floor, so what comes back is whatever was already in the room. Ask what would make someone abandon this in the first week and the space closes. That question has a small number of answers, they're specific, and most people present can name two of them without preparing.

Success has too many causes to rank. When something works you can rarely say which of the twelve things you did was the one that carried it, and the story you tell afterwards is mostly reconstruction. Failure is narrower. The same five causes come back project after project, and they're usually known. What stops them being said is that the meeting was billed as a growth meeting, and in a growth meeting pessimism sounds like disloyalty.

There's no template to fill in and no workshop to book. You need one sentence and twenty minutes of tolerance for an unflattering list. Tolerance is where it gets difficult, because inverted questions are rude. Writing what makes this worse than the thing it replaced on a wall, in front of the people who built it, takes some nerve. It also gets answers the polite version never does.

It pairs well with force-field thinking. Name the goal, write down what's pushing toward it, write down what's pushing against it, then put your effort into the second list. Clearing an obstacle is usually more tractable than manufacturing an advantage, and often enough the obstacle turns out to be something you built.

IN PRACTICE

Questions worth putting on a page before a roadmap review. Each is the inverted twin of one that produced nothing:

- What makes it hard for a new customer to become a customer at all?
- What makes someone abandon paying a bill in the app and finish it on the website instead?
- What makes the experience after a purchase worse than the purchase itself?
- Why would someone who used the marketplace once decide not to come back?
- What are we doing that quietly erodes the reason to open the app at all?
- If we wanted this launch to be ignored, what would we do? Are we doing any of it?
- What would make the team stop trusting this metric?

WHERE IT BREAKS

Inversion tells you what to avoid. It never tells you what to build. A product can have every reason to leave removed and still give nobody a reason to arrive, and a team that runs on this alone gets very good at being inoffensive. It also degrades when the failure modes are genuinely unknown: a new market, an unfamiliar technology, a user nobody has met. The list you produce there is really a list of your existing fears. Use it once you have a direction, not to find one.

Charlie Munger, after the mathematician Carl Gustav Jacobi. Before You Commit treats this as a stage in a sequence rather than a standalone tool.

```
by analogy ▸ X → X′ → X″
from parts ▸ ● ● ● → ?
── one copies, one asks ──
```

02 First Principles

Take it apart until you reach something that's actually true, then build back up without carrying the old answer.

Reasoning by analogy is the default, and there's nothing wrong with it most of the time. It's fast and the last answer is usually close enough. What it does badly is inheritance. Every copy carries the constraints of the thing it copied, including the constraints that stopped being real years ago. You end up with a product shaped by a decision from a meeting nobody remembers. Nobody defends it, because nobody knows it's there. It arrived as furniture rather than as a choice.

Working from first principles means refusing that inheritance on purpose. Break the problem into pieces small enough that each one is either a fact or a decision. Facts you can check. Decisions you can interrogate: who made this, what were they optimising for, is that still what we're optimising for. None of it is clever. It's tedious, and the tedium is why people skip it.

Musk's semantic tree is a good image for it. Understand the trunk and the big branches before you spend time on leaves, otherwise there's nothing for the leaves to hang from. In a working session that means resisting the pull toward implementation in the first hour. Which component library is the wrong question. What does this screen have to be true about is the right one, and it's harder, which is presumably why the first one gets asked.

There's a cost discipline attached that people tend to drop. Decomposing a problem is not a licence to rebuild it. Once you've taken it apart you still have to price rebuilding against iterating, in time, attention and risk, and the method has nothing to say about that. It shows you what's possible. Worth it is a separate question and you have to answer it yourself.

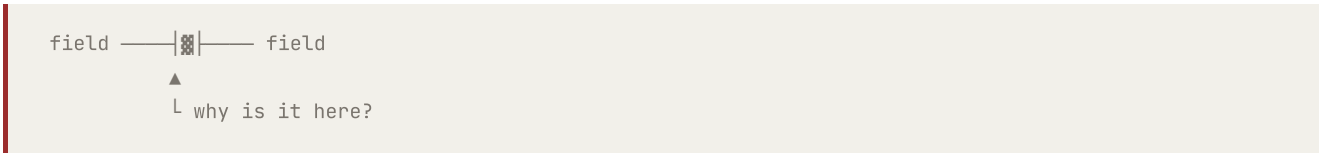
IN PRACTICE

Questions that pull a problem back to its trunk:

- What are the smallest pieces of this that are either a fact or a decision?
- Which of these did we inherit rather than choose?
- Who made that original choice, and what were they optimising for?
- What has changed since then that would have changed their answer?
- If this did not exist and we were building it today, what would we build?
- What are we treating as a constraint that is actually a preference?
- What does this cost to rebuild, honestly: time, attention, and the risk of ending up with something worse than we have?

> WHERE IT BREAKS
>
> It's slow, and the slowness has a price nobody counts. Most decisions don't deserve it. For the ninety percent that are small and reversible, reasoning by analogy is simply correct. It's also the favourite tool of anyone who wants to reopen a settled question, which is how teams end up rebuilding their authentication for the third time. And decomposition has a floor. Below a certain level of detail you're not reasoning from principles any more, you're guessing with extra steps. If you can't check a fact, it isn't a first principle. It's an assumption in better clothes.

Aristotle, by way of Elon Musk's rendering of it.



03 Chesterton's Fence

Before you remove something, be able to say why it was put there. If you can't, you're not ready.

Chesterton's example is a fence across a field with no obvious purpose. The reformer says he can see no use for it and proposes clearing it away. The better answer is: go and find out why it's there, come back and tell me, and then I might let you destroy it. This isn't an argument for conservatism. It's a point about evidence. Not knowing why something exists is a fact about you, not about the thing.

Product work is a field full of fences. A validation rule nobody can explain. A step in a flow that adds friction and has somehow survived three redesigns. A feature with terrible usage numbers that a small, quiet group depends on completely. All of them look removable. Some of them are load-bearing in a way that only becomes visible two quarters later, when a regulator or a partner or four hundred users arrive to explain it.

In practice it comes down to something short. Find one person who was there. Documents won't help much, because they record what was intended rather than what was learned. Fences generally go up after an incident, and the incident lives in somebody's memory, often in a different team, sometimes at a company they left. Fifteen minutes of asking around turns up reasons that no amount of code reading would.

There is an opposite failure worth naming. Some fences are genuinely pointless, and treating all of them as sacred is how a product turns into an accumulation of other people's fear. The rule isn't don't remove. It's don't remove blind. Once you can state the reason, you've earned the right to decide the reason no longer holds.

IN PRACTICE

Questions to ask before deleting anything that already works:

- Who added this, and when?
- What happened in the weeks before it was added?
- Is there an incident, a complaint, or a contract behind it?
- Who would notice if it disappeared tonight, and how long would it take them to notice?
- Is the usage low because nobody needs it, or because the people who need it are few and quiet?
- What is the reversal cost if we remove it and we were wrong?
- Can we make it removable before we make it removed, by turning it off for a slice and watching?

WHERE IT BREAKS

It gets used in bad faith constantly. Demand a full archaeology for every change and nothing ships, and why is this here becomes the cheapest way to block work you dislike without having to argue against it. It also assumes there's an origin to find. In a system old enough, a good share of the fences were built by people reasoning exactly the way you are now, so the reason is real but it's somebody else's misunderstanding, faithfully preserved. Put a time limit on the search. If nobody can produce a reason within a day, that's your answer.

G. K. Chesterton, The Thing, 1929.

```
two-way ▸ decide fast, walk back
one-way ▸ slow down, you're stuck
── most doors are two-way ──
```

04 Reversible and Irreversible Decisions

Work out which kind of door you are standing in front of, then match the process to it.

Bezos put it as two types, and the point is that organisations routinely apply the wrong process to each. A two-way door you can walk through, look around, and come back out of if you don't like it. A one-way door you can't. Firing someone. Launching publicly. Signing a five-year lease. Choosing the data model everything else will depend on.

The failure runs both ways and the second direction is more common. Big companies put heavyweight process on two-way doors, which is how an experiment that costs a week ends up needing four approvals. Small ones sprint through one-way doors

because sprinting is the culture, then find out eighteen months later that a schema decision made in an afternoon takes a quarter to undo.

The question that sorts them isn't how important is this. It's what does reversal cost. Importance and reversibility come apart constantly. A pricing experiment can matter enormously and reverse in a day. A choice of authentication provider can feel like plumbing and turn out to be the thing you can never get out of.

Most doors are two-way, and most teams behave as though most doors are one-way. That gap is where the speed is. Classify honestly and you get to move fast on the majority and slow on the few that have earned it. Classifying takes about a minute and almost nobody does it out loud.

IN PRACTICE

Questions to classify a decision before deciding how much process it deserves:

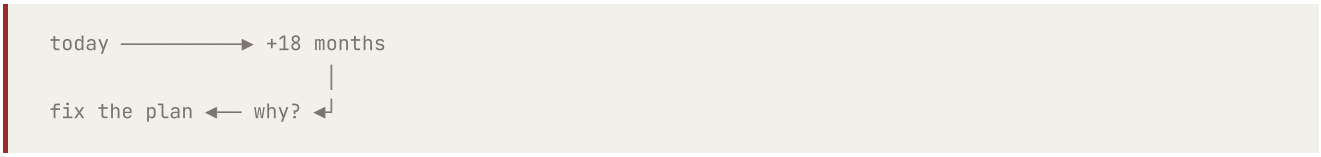
- If this goes badly, how long does undoing it take and what does undoing cost?
- Who or what gets locked in: data, contracts, users, other teams' work?
- Are we treating this as irreversible because it's important, or because it is?
- What is the cheapest reversible version of this same decision?
- If it's a two-way door, why does it need this many approvals?
- If it's a one-way door, has anybody actually said so out loud?
- What would make this reversible that isn't true today?

> WHERE IT BREAKS
>
> The classification is easy to get wrong in whichever direction suits you. Anything can be argued into being a two-way door by somebody who wants to move, and reversal costs are always underestimated because they get paid later and usually by someone else. Some decisions are also technically reversible and socially not. You can roll the feature back, but you can't unannounce it, and you can't rehire the person. Reversibility of the artefact is not reversibility of the situation.

Jeff Bezos, 2015 Amazon shareholder letter, as Type 1 and Type 2 decisions. Before You Commit runs this as a triage step, with the process weights attached.

II Attacking the idea

You have a direction. Now try to kill it on paper, while killing it is still cheap.



05 Pre-Mortem

Move to the day the project failed and write the autopsy while there's still time to prevent it.

A retrospective arrives after the damage and produces lessons for a project that's already finished. A pre-mortem runs before anything is committed, while the plan is still soft enough to change. All of it hangs on one instruction: it's eighteen months from now, this failed completely, write down why. Not could fail. Did fail. The grammar matters more than it sounds like it should.

Klein's work on this is about prospective hindsight. Imagining an outcome as already settled improves people's ability to generate its causes by roughly thirty percent, compared with asking how likely it is. Part of that is cognitive and part of it is social. Asking what could go wrong makes you sound negative in front of the sponsor. Saying it failed, and here's my theory of why, makes you sound useful. Same content, different standing.

That social half is where most of the value sits. Almost every team has somebody who already knows the answer. The engineer who integrated with this provider at their last job. The support lead who has watched this flow confuse people for a year. The designer who was told to cut the state that's about to come back as a bug. The exercise gives them a licensed occasion to say so without looking like they're against the project.

Run it silently first. Everyone writes for five minutes alone, then the causes get read out. Skip that and the first person to speak sets the frame, and the room quietly converges on their failure mode. Then take the top three and change the plan for each one before you leave. A pre-mortem that produces a risk register nobody opens again is theatre.

IN PRACTICE

Questions from a pre-mortem on a third-party payment integration. The shape generalises to any dependency you don't control:

- What do we do if the design work does not arrive on time?
- What is our fallback when the test environment is down for a day, or for a week?
- If the provider changes its business rules after launch, how fast can we adapt?
- Did we build on their components? If those change, do our screens and flows break?
- What happens to us if their infrastructure goes down entirely?
- If we miss the deadline, what is the actual plan B, and who decides to trigger it?
- If performance is worse than promised, what is the alternative and at what point do we switch?
- If stored card details change format later, does that become our problem?
- What if two payment paths fail at the same time?

> WHERE IT BREAKS
>
> It surfaces the failures your team can already imagine, which leaves it blind to the ones nobody in the room has lived through. That makes it strongest on a familiar project and weakest on a novel one, which is precisely backwards from where you'd want it. Timing catches people too. Run it before the plan is concrete and you get generic anxieties like the market might change, and there's nothing to do with those. The window is after there's a real plan and before there's a real commitment. And watch the output. If what you produced is a document rather than a changed plan, nothing happened.

Gary Klein, developed from research on prospective hindsight and popularised in the Harvard Business Review. Before You Commit builds a full stage around this, with a template.

```
the idea  ●
          |◂ what kills this?
          |◂ what must be true?
          |◂ what can't we say?
```

06 Devil's Advocate

Appoint someone whose only job is to destroy the idea, and protect them while they do it.

The devil's advocate was an actual office. The *advocatus diaboli* was appointed by the Church to argue against a proposed canonisation, and it existed because an institution that only ever hears the case for something will eventually canonise anyone. What's

worth stealing is the structure. Scepticism has to be somebody's assigned job. It can't be a personality trait you hope is present in the room.

Being assigned is what makes it survivable. Objecting to a senior person's idea on your own account is a career move. Objecting because you were told to, in a role everyone can see is a role, costs nothing. An identical sentence lands differently depending on whether it came with a mandate. So say the name out loud when you run one of these: for the next twenty minutes she's trying to kill this, and none of it is personal.

Aim the attack at load-bearing assumptions rather than surface details. Anyone can find a weak sentence in a deck. One question earns its keep: which single belief, if false, brings the whole thing down. Then: what evidence do we have for it, how good is that evidence, what would we expect to see if it were wrong. Most plans rest on one or two beliefs like this and nobody has said which ones.

An appointed advocate underperforms a real one, and this is well documented. People asked to argue a position they don't hold argue it politely, and everyone discounts an argument they know is performed. Two repairs. Rotate the role so nobody becomes permanently the difficult one. And when the stakes justify it, find someone who actually disagrees and ask them to make their real case, rather than asking a supporter to pretend.

IN PRACTICE

Questions the role exists to force into the room:

- Which single assumption, if false, kills this entirely?
- What evidence supports that assumption, and how strong is it really?
- What are the limits of that evidence? What does it not cover?
- What would we expect to see if we were wrong, and are we looking for it?
- Who benefits from this being approved, and are they the ones presenting it?
- What is the strongest version of the case for doing nothing?
- If a competitor announced exactly this tomorrow, what would we say is wrong with it?
- What are we not allowed to say in this meeting?

> WHERE IT BREAKS
>
> Ritualised dissent stops being dissent. Once it's a fixed slot in every review the room learns to sit through it, and objections come back answered with prepared lines rather than thought. There's a specific abuse to watch for: a leader who has already decided runs the exercise to demonstrate that objections were heard, which does more damage than never asking. It can also burn the person doing it. If the same individual is always the one attacking, they end up with exactly the reputation the role was supposed to protect them from.

The Catholic Church's *advocatus diaboli*, an office abolished in 1983.

```
claim
  └▸ what do you mean?
      └▸ what must be true?
          └▸ what breaks it?
```

07 Socratic Questioning

Don't answer the claim. Take it apart with questions until the assumption underneath is visible.

People use the Socratic method as a debate tactic, which is a waste of it. What it's for is moving a conversation from positions to reasons. Two people arguing about whether to build something can go round for an hour. The same two asking what would have to be true for this to be worth building usually locate the disagreement in ten minutes, and it's almost never where either of them thought.

There are four movements and the order matters. Clarify first. What do you mean, can you give an example. A surprising share of disagreement turns out to be two people using one word for two things, and you can end a meeting right there. Then assumptions: what has to be true for that to hold. Then other angles: what's the counter-case, why this rather than the obvious alternative. Then consequences: if everyone believed this, what would follow.

Staying in questions is the discipline. The moment you turn one into a statement, so what you actually mean is, you've taken the position away from the other person and they'll defend it instead of examining it. Questions leave it where it belongs. This is also why the method works upward. You can question a decision made three levels above you without contradicting anybody, which is occasionally the only move available.

Turned on your own thinking it's slower and worth more. Write down the claim you believe about the product, then ask in writing what would have to be true for it, and answer honestly. Most strategy documents contain one sentence that has never been examined. It's usually in the first paragraph.

IN PRACTICE

The four movements, as questions you can use verbatim:

- Clarify: what do you mean by that? Can you give a concrete example?
- Clarify: is this always true, or true in the cases we have looked at?
- Assumptions: what has to be true for this conclusion to hold?
- Assumptions: what are you taking as given that we have not checked?
- Alternatives: what is the strongest counter-argument?
- Alternatives: why this rather than the obvious other option?
- Consequences: if everyone acted on this, what would happen?
- Consequences: what does this commit us to that we have not said out loud?
- Evidence: what would change your mind?

WHERE IT BREAKS

It's exhausting to receive, and it reads as hostile long before it feels hostile to the person asking. Pointed at a person rather than a claim it becomes interrogation, and the honest answer, I don't know, gets punished. Do that a few times and everyone arrives with defences prepared instead of thinking. It also fits badly with time pressure. Some decisions need a call rather than an examination, and a leader who Socratises their way through an incident isn't being rigorous. They're avoiding the decision.

Socrates, via Plato. The four-movement framing is the standard modern teaching version.

```
○ facts   ○ upside   ○ risk
○ feeling ○ options  ○ process
── one at a time, together ──
```

08 Six Thinking Hats

Have the whole room think in the same mode at the same time, and switch modes on purpose.

An ordinary meeting runs every mode at once. Someone brings data, someone brings enthusiasm, someone brings objections, someone brings a feeling they can't justify. Because it's all happening simultaneously, the discussion stops being about the idea and becomes about who wins. De Bono's answer is parallel thinking. Everyone wears the same hat at the same time, and the hat changes when the chair says so.

The six modes are information (what do we know, and what do we only believe), benefits (the upside, put as strongly as it can be put), risk (what fails, and what that costs), feeling (the gut reaction, allowed without justification), creativity (alternatives

and options nobody has raised), and process (what are we doing, what's next, are we done). The last one belongs to the chair and nobody else.

Two of them do most of the work. Feeling is the one people cut first and the one that most often holds the real objection. Someone senses a plan is wrong twenty minutes before they can say why, and if the meeting has nowhere to put that, the signal either disappears or comes back disguised as a technical concern that doesn't quite hold up. Risk is the other one. Give it a slot of its own and it stops leaking into all the others.

What you actually get out of it is time. Ten focused minutes per mode beats an hour of unstructured argument often enough to be worth the awkwardness. It also flattens the room's built-in bias. The cautious person no longer has to fight for airtime, because there's a segment where caution is the only permitted contribution and another where it isn't permitted at all.

IN PRACTICE

Running it on a real decision, as prompts per mode:

- Information: what do we know, what do we only believe, and where is the gap?
- Information: what data would settle this, and can we get it this week?
- Benefits: state the best possible outcome as though it will happen.
- Risk: what fails, how likely, what does it cost, and can we detect it early?
- Feeling: what is your gut reaction? No justification required or accepted.
- Creativity: what is a third option nobody has proposed?
- Creativity: what would we do if this budget were doubled, and if it were halved?
- Process: what did we just decide, what is still open, and who owns it?

> WHERE IT BREAKS
>
> This is a facilitation technique and it needs a facilitator. With nobody enforcing the mode the room reverts inside five minutes and you've added ceremony to the same argument. It suits small groups badly, since with three people parallel thinking is just three people doing an exercise. It suits genuinely adversarial situations worse, because there the disagreement is about interests rather than analysis and no amount of structure touches that. Mostly it risks becoming a costume. Teams announce the hats and then hold the normal meeting, with worse vocabulary.

Edward de Bono, Six Thinking Hats, 1985.

III Following the consequences

The first effect is the one everybody plans for. The second shows up in the numbers a year later.

```
act → 1st → 2nd → 3rd
        ▲     ▲
  obvious    where it lands
```

09 Second-Order Thinking

And then what? Ask it three times. The consequence that matters is rarely the first one.

First-order thinking asks what happens if we do this. It's fast, it's usually right about the immediate effect, and it's the level at which nearly every decision actually gets made. Second-order thinking asks and then what. What do people do in response to that effect, and what follows from what they do. Marks's point is that the first level is where everybody already is, so there's nothing left to be gained there.

The subsidised-loan example is the standard one and it still holds up. Society wants more people in higher education. Many can't afford tuition. Subsidise the loans and the first-order effect arrives as intended: more people can go. Then institutions discover that tuition can rise without demand falling, because the subsidy absorbs the difference and eighteen-year-olds are not well placed to price thirty years of debt. Tuition more than doubles in real terms. Every step in that chain was reasonable and the outcome is the opposite of the intent.

Ecology has the short version. These toads will eat the pests we hate is first-order. These toads are poisonous, have no local predators, and will shortly be the pest is second-order. That shape recurs: the first effect is mechanical, the second is behavioural, and behaviour is where you get surprised.

In product work the behavioural layer is most of what matters. Ship a feature and the first-order effect is that people can now do the thing. Then they change how they use everything around it. Support inherits a new class of question. A metric somebody is measured on moves for reasons unrelated to anything they did. A team starts optimising for the new number. None of that is in the spec.

IN PRACTICE

Questions to run before committing, in order:

- What is the intended effect, stated plainly?
- Who changes their behaviour in response to that effect?
- And then what happens? Ask again.
- And then what happens after that?
- Which of our own teams starts optimising for something new because of this?
- What does this make cheap that used to be expensive, and who exploits that first?
- What does support inherit that they were not staffed for?
- Who is affected who was not in the room when we decided?
- In a year, what will we say the real consequence was?

> WHERE IT BREAKS
>
> There's no natural stopping point, and by the third or fourth order you're speculating in the tone of analysis. A team that gets good at this can talk itself out of anything by inventing a harm four steps down. It also pushes toward inaction, because harms are easier to imagine than benefits. The second-order upside is just as real and gets a fraction of the airtime. Two steps, done honestly, for decisions that are hard to reverse. One step for the rest.

Howard Marks, The Most Important Thing. The framing is older. Hazlitt made the same argument about economics in 1946. There is a longer treatment of this, with a worked example, in Before You Commit.

```
proxy ≈ what you want
  └ make it a target →
      proxy ↑   thing ↓
```

10 Goodhart's Law

When a measure becomes a target, it stops being a good measure.

Goodhart wrote it about monetary policy. Strathern gave it the sentence everybody quotes. Nothing about the mechanism requires anyone to behave badly. A metric is a proxy: something narrow and measurable that correlates with something wide and unmeasurable that you actually care about. The correlation holds as long as nobody is pushing on it. Attach consequences and people optimise the proxy directly, along exactly the paths where it comes apart from the thing it was standing in for.

Calling it cheating is how organisations miss what's happening. A team measured on tickets closed that closes tickets faster is doing the job as defined. A team measured on weekly active users that ships a notification bringing people back for four seconds

is doing the job as defined. The failure sits in the definition, and whoever wrote the definition is usually the last to notice, because from where they sit the number is going up.

Worth carrying around: every metric you publish is an instruction. Putting a number on a dashboard, into a review, or into somebody's objectives changes behaviour whether you meant it to or not. So the question before promoting a metric isn't whether it's accurate. It's what people will do to move it, and whether you're happy with all of that.

There are partial defences and none of them is a cure. Pair each metric with a guard that isn't allowed to move the wrong way: activity with retention, speed with reopened tickets, conversion with refunds. Rotate what you emphasise so no single proxy has long enough to be gamed into meaninglessness. And keep one thing you look at that nobody is measured on. It stays honest for exactly that reason.

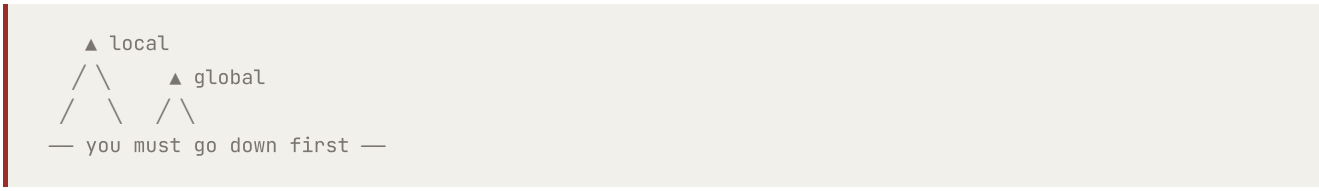
IN PRACTICE

Questions before you put a number in somebody's objectives:

- What do we actually care about, and how is this number standing in for it?
- What is the cheapest way to move this number without doing the underlying thing?
- Would we be happy if someone moved it that way?
- Which guard metric must not get worse while this improves?
- Who is measured on this, and what will they stop doing to make time for it?
- How would we notice if the correlation had broken?
- Is there a number here we look at but nobody is judged on?
- When do we retire this metric?

> WHERE IT BREAKS
>
> Goodhart describes a risk, not a certainty. Cited broadly enough it turns into an argument against measuring anything, which costs far more than the failure it's protecting against. Unmeasured teams don't become wise. They become unaccountable and drift toward whatever is most pleasant to build. All metrics decay, and the answer is maintenance rather than abstinence. Expect the proxy to be gamed, watch for it, and replace it before it's meaningless rather than after.

Charles Goodhart, 1975. The familiar wording is Marilyn Strathern's, 1997.



11 Local and Global Optimum

A team can improve its own number and make the whole thing worse. That is the normal case.

An optimisation problem can have a peak you reach by always walking uphill and a higher peak elsewhere that you can only reach by walking down first. Hill-climbing finds the near one and stops, because from up there every direction looks worse. Nothing about standing on a local optimum feels like being stuck.

Organisations are hill-climbers by construction. Each team is measured on its own slope and each improvement is real. Support cuts resolution time by deflecting tickets into a help centre, and the deflected tickets are the ones that would have told product what was broken. Everybody climbed. The mountain got shorter.

That's what makes these hard to argue about. There's no villain and no mistake to point at, every step was locally correct, and the sum is worse. So the conversation has to happen at a level where somebody owns the whole surface, and usually nobody does.

Getting off one costs something visible and pays something invisible, which is the worst possible shape an organisational decision can have. You accept a measurable drop now for a hypothetical gain later, and whoever proposes it owns the drop personally. Teams sit on local optima for years over this alone, not because nobody noticed.

IN PRACTICE

Questions for when everything is improving and nothing is getting better:

- Which number is each team climbing, and do they add up to what we want?
- Where does one team's improvement arrive as another team's problem?
- What would we have to make worse in the short run to get somewhere better?
- Who owns the whole surface, and are they in this conversation?
- Are we hill-climbing because it's right, or because every step has to look like progress?
- What did we stop being able to see when we optimised this?
- If we rebuilt this system from scratch, would we build what we have?

> **WHERE IT BREAKS**
>
> The model implies a global optimum exists and can be found, and usually neither holds. What you get in practice is a claim that some other unspecified peak is higher, which is unfalsifiable and a convenient argument for any large rewrite. It also underrates hill-climbing, which is how most real improvement happens and is far safer than jumping. Use it to explain why a set of genuine local wins produced a worse whole. Treat any specific proposal to climb down as a bet that needs its own case.

Optimisation theory. The organisational reading is standard in systems thinking.

IV Judging under uncertainty

You won't have enough information. These are about deciding anyway, and doing it honestly.

```
this one is different?
──────────────────────
last ten:  ▓▓▓▓▓▓▓▓▓▓
seven slipped
```

12 Base Rates

Before you estimate this case, find out what happened in the last ten like it.

Given a specific case, people reason from its details and ignore how often things of that kind turn out each way. This team is strong, the scope is clear, we've done the prep, therefore eight weeks. A base rate asks something duller. Of the last ten projects that looked like this at this stage, how many finished in eight weeks? The answer is usually available and usually worse, and it usually gets ignored anyway.

Kahneman calls these the inside view and the outside view. The inside view is built from the specifics of your case, and it feels like knowledge because you have so much of it. The outside view treats your case as one member of a class and asks what the class does. When the two disagree, the outside view wins more often than anyone wants to accept. Experience doesn't close the gap either. Experienced people just have more specifics to be confident about.

Choosing the reference class honestly is where it gets hard. Every project has something that makes it unlike the others, and that something is the escape hatch. This time is different because we're on a new framework, because the team has changed, because we planned better. Occasionally that's true. Most of the time the difference is real and irrelevant, and the class you should be using is the boring one: projects of roughly this size, in roughly this system, with roughly this many dependencies you don't control.

It works best as a starting number rather than a verdict. Take the base rate, adjust for the specifics, and make the adjustment something you argue for rather than assume. A forecast that starts at the historical average and moves twenty percent for a stated reason is a different animal from one that starts at optimism and then gets defended.

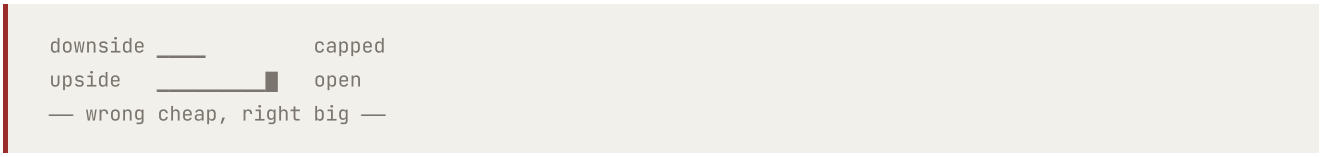
IN PRACTICE

Questions to force the outside view into an estimate:

- What is the reference class here? What are the last ten things that looked like this?
- How did those actually go, in dates rather than in memory?
- What is the historical rate, before we discuss this case at all?
- What specifically is different this time, and is that difference relevant to duration?
- Have we said this same difference before, on a project that then slipped?
- Which parts of this estimate depend on someone outside our control?
- What is our track record with that dependency specifically?
- If I had to bet my own money on the eight-week date, would I?

> WHERE IT BREAKS
>
> It needs a reference class, and often there isn't a real one. A genuinely new product, a market nobody has entered, a first integration with a partner nobody has worked with. Force a class into existence there and you get a confident number derived from irrelevant history, which is worse than admitting you're guessing. It's also a poor tool for anything you're deliberately trying to make an outlier. Taken as law it says most attempts fail, therefore don't attempt, and that isn't a decision procedure. It's a way of never doing anything unusual. Use it to price the risk rather than to refuse it.

Daniel Kahneman and Amos Tversky. The inside-view framing is from Thinking, Fast and Slow.



13 Expected Value & Asymmetry

Judge the decision by the distribution you were facing, not by the outcome you happened to get.

Expected value is boring arithmetic: each outcome weighted by its probability, then summed. That number is rarely the useful part. What's useful is that you can't compute it without writing down your probabilities and payoffs, and writing them down is where the disagreement in a room becomes visible. Two people who can't agree on a decision usually turn out to disagree about one probability that neither of them had said out loud.

What follows from it is harder to absorb than the arithmetic. A good decision can lose. A bad one can pay. Outcome and decision quality are separate variables, and an organisation that judges only outcomes trains its people to avoid any bet that can visibly fail, which is every bet worth taking. After a loss the question worth asking is whether you'd take the same bet again with the same information, not what went wrong.

Asymmetry is where the practical value is. The good decisions available to most teams aren't the ones with the highest expected value. They're the ones where the downside is capped and the upside isn't: an experiment that costs a week and might change the roadmap, a small bet on a channel that either does nothing or works, an architecture choice you can walk back. You don't need to be right often when being wrong costs a week and being right changes the year.

Then there is the capped-upside trap, where the best case is that nothing happens and the worst case is severe. A risky migration with no user-visible benefit. A dependency that saves a fortnight and can take the product down. The asymmetry runs the wrong way and these get approved constantly, because the upside is legible on a plan and the downside is only a probability.

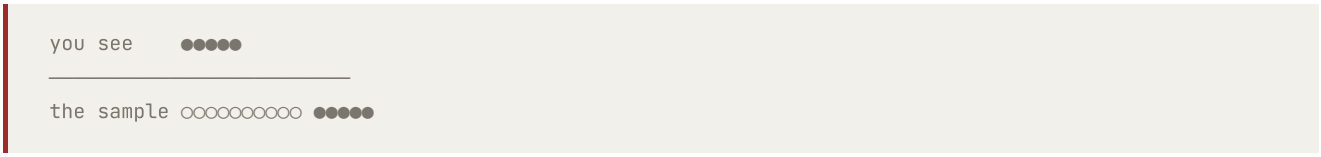
IN PRACTICE

Questions to make the shape of a bet visible:

- What are the possible outcomes, and roughly how likely is each?
- What is the worst case, in money, in time, and in trust?
- Is the downside capped? By what, specifically?
- Is the upside capped? If it is, why are we doing this?
- What is the smallest version of this bet that still teaches us something?
- If this fails, do we find out in a week or in a year?
- Would I take this bet ten times in a row?
- Are we judging the last decision by its outcome, or by what we knew at the time?

> WHERE IT BREAKS
>
> The probabilities are made up. Expected value lends the authority of arithmetic to numbers somebody guessed, and a decision defended with a computed figure is harder to argue with than one defended by instinct, even when the instinct was better founded. It also assumes repetition. An average only means something over many trials, and a bet you take once with a ruinous tail isn't made safe by a favourable expected value. For anything that can end the project, the question isn't the mean. It's whether you survive the left tail.

Classical probability. The decision-versus-outcome framing is Annie Duke's, in Thinking in Bets.



14 Survivorship Bias

You're studying the ones that made it. The lesson is in the ones you can't see.

Wald's aircraft are the canonical illustration. Returning bombers had bullet holes concentrated in the fuselage and wings, and the reasonable conclusion was to armour where the holes were. Wald said the opposite. Armour where there are no holes, because those are the places that, when hit, meant the plane didn't come back. The data was a sample of survivors, and it was silent about the failures in the most misleading way available, which is by omission.

Most product wisdom has this shape. The companies writing the blog posts are the ones that survived, and the practices they credit were also in use at the companies that didn't, which wrote nothing. The user research you can run is with people still using the product. The retrospective covers projects that shipped. Every one of those samples is survivors, and the causal story you build from it has no control group.

Correcting for it isn't clever, it's just work. Go and find the absences on purpose. Interview churned users as well as active ones. Read the post-mortems of failed projects inside your own company before you read case studies of successful ones elsewhere. When somebody credits an outcome to a practice, ask how many teams did the same thing and aren't around to say so.

It persists because survivors are cheap and non-survivors are expensive. Active users answer surveys. Churned users answer nothing. Successful projects have documentation. Cancelled ones have a folder nobody opens. This bias is structural, built into who is available to talk to, so being aware of it changes nothing on its own. Only the expense of going after the missing half does.

IN PRACTICE

Questions to find the sample you are missing:

- Who is not in this data, and why are they not in it?
- Are we researching with people who stayed? What did the people who left say?
- What happened to the teams and companies that tried this and are not around to write about it?
- Is this a practice that causes success, or one that successful and unsuccessful teams both used?
- What does our own list of cancelled projects say, and has anyone read it?
- If this had failed, would we have heard about it?
- What is the cheapest way to get one honest conversation with someone who abandoned us?

> WHERE IT BREAKS
>
> Missing data stays missing, and that isn't a solvable problem. You can reduce the bias. You can't remove it. Pushed hard the model turns into unfalsifiable scepticism, where any evidence gets dismissed with but what about the ones we can't see, and then you never update on anything. There's a real cost too. Chasing churned users is expensive and low-yield, and for small decisions it isn't worth it. Save the effort for conclusions you're about to build a year of work on.

Abraham Wald, Statistical Research Group, 1943.

15 Power Laws

Most of the effect comes from very few of the causes, and the average is the wrong summary.

In a normal distribution the average describes almost everybody and the extremes are rare. In a power law the average describes almost nobody. A few items carry most of the total and the rest form a long tail that's individually negligible and collectively large. Feature usage, revenue per customer, session length, bug frequency and outage cost all sit closer to the second shape.

Which makes the mean actively misleading. Average session length in an app where two percent of users account for sixty percent of the time is a number describing no user who exists. Percentiles say more, and the useful ones are at the ends: your p95 latency is where the complaints come from, and the top one percent of accounts is where the revenue is.

Two opposite mistakes follow. One treats the head as the whole product and lets the tail rot, which is how you lose the market that eventually matters. The other spreads effort evenly across a long list because every row looks equally deserving, when a fifth of the list carries four fifths of the value.

The shape governs bets too. If outcomes are power-law distributed, most attempts return roughly nothing and a very few return everything, so the number of attempts matters more than the average quality of one. That reasoning is right for a venture portfolio and wrong for most roadmaps, where you never get enough attempts for the tail to show up.

IN PRACTICE

Questions before acting on an average:

- Is this distribution roughly normal, or does a small share carry most of the total?
- What does the mean hide here, and what do the percentiles say instead?
- What fraction of users, features or accounts produces most of the value?
- Are we spreading effort evenly across things that are not evenly valuable?
- Where is our p95, and who lives there?
- How many attempts would a long-tail strategy need, and do we get that many?
- What is in the tail that we have stopped looking at?

> WHERE IT BREAKS
>
> Power law gets asserted far more often than it gets checked. A skewed sample is not a power law, and the eighty-twenty split is quoted about datasets nobody plotted. It also gets used to justify neglect: the tail is small, so ignore it, which is backwards in markets where the tail is where new demand appears. And a distribution that looks power-law across the range you can see may not hold outside it, which is exactly where the expensive decisions get made.

Pareto, through a long line of restatements. The venture reading is Peter Thiel's, in Zero to One.

```
decide now   ▸ ■ one path
decide later ▸ ■■■ still three
── what does waiting cost? ──
```

16 Preserving Optionality

Some decisions get cheaper if you wait. Work out which, and refuse to make those early.

A choice made before it has to be made costs you every option it closes and buys nothing except the feeling of having decided. Optionality is the discipline of noticing when that's happening. It isn't indecision, which is failing to decide when it's time. This is deciding when to decide, and treating that as a question of its own.

The mechanics are simple enough. Find the last responsible moment, the point where waiting starts costing more than deciding. Before it, information is still arriving and every week makes the decision better. After it, you pay for delay in coordination, rework and people waiting. Most teams can't say where that line falls for any given decision, so they land on both sides of it at random.

In product work it shows up as commitment nobody asked for yet. Choosing the vendor in month one for a launch in month nine. Building a schema for requirements still moving. Naming the thing publicly before you know what it is. None of those had to happen when they happened, and each closed doors that were free to leave open.

The counterweight is path dependence. Options don't stay open by themselves, they decay. Pricing changes, the team forgets the context, and a run of small local decisions quietly rules out the big one you thought you were preserving. Tyranny of small decisions is the name for that. Keeping an option open is active work.

IN PRACTICE

Questions before committing to something you don't have to commit to yet:

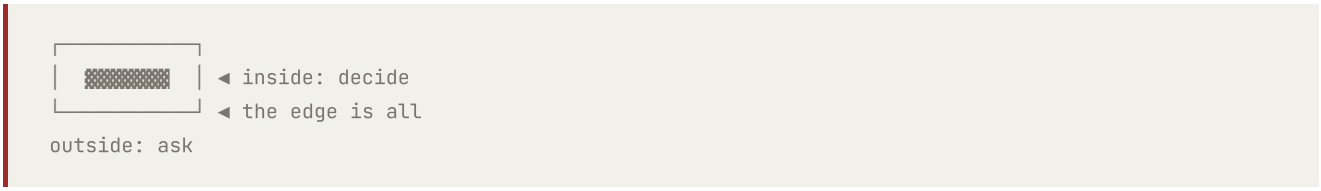
- What is the last responsible moment for this, and what changes after it?
- What information arrives between now and then?
- Which options does deciding now close, and how expensive are they to reopen?
- What does waiting actually cost, in coordination and rework?
- Can we make a reversible version of this instead of the real one?
- Which small decisions are quietly ruling out the option we think we're keeping?
- Are we deciding early because we have to, or because deciding feels like progress?

> **WHERE IT BREAKS**
>
> Used badly this is a licence never to decide, and it arrives with its own justification attached, which makes it more dangerous than ordinary procrastination. Options carry costs that rarely get counted: holding two paths open often means building for both, and that can cost more than picking the worse one and committing. Teams that hold everything open also leave everyone else nothing to plan against. The question is never whether to preserve optionality. It's which specific option is worth what keeping it costs.

Real options theory. The software framing comes from lean and agile practice, where it appears as the last responsible moment.

V Being honest with yourself

The last three aren't about the problem. They're about you.



17 Circle of Competence

Know where your judgement is worth something. The edge matters more than the size.

Munger's version is that you don't have to be an expert in everything, but you do have to know precisely where your expertise stops. The circle can be small. Small circles are fine, and often more profitable than large ones. What matters is that the boundary is sharp, because the serious errors happen just outside it, in the band where you know enough to have an opinion and not enough to know the opinion is bad.

Ignorance isn't the danger. Total ignorance is safe, because you ask somebody. Adjacent competence is the dangerous zone. A product manager who has shipped six consumer features, forming a confident view about a regulated payments flow. An engineer who scaled one system, reasoning by analogy about a different one. Confidence transfers across the gap and competence doesn't, and from the inside the two feel identical.

A practical test asks about consequences rather than credentials. If I'm wrong about this, who pays, and can they recover? Inside the circle you act on your own judgement. At the edge you still act, but you build in a check: a review, a small first version, somebody who does know. Outside it you don't decide at all. You find the person who should and give them what they need to.

The circle moves in both directions. It grows where you do the work. It shrinks in areas you haven't touched in three years, and it shrinks quietly while your confidence stays where it was. That's the one that catches people. There's no signal for it. You feel the same as you did when you knew.

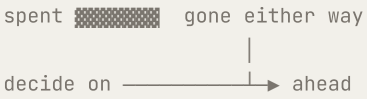
IN PRACTICE

Questions to locate the edge before you cross it:

- Can I explain how this works to someone who does it for a living, without hedging?
- When did I last do this myself, rather than manage someone doing it?
- If I am wrong here, who pays, and can they recover?
- Am I confident because I know this, or because it resembles something I know?
- Who in the building actually knows, and have I asked them or just informed them?
- What would the expert say I am about to get wrong?
- Is this a decision I should make, or one I should stage and check?

> WHERE IT BREAKS
>
> It's easy to use as an excuse. Staying inside a small circle is comfortable, and a leader who reaches for this too readily stops making the cross-domain calls that are the job. Most senior decisions are, by construction, outside everybody's circle. It's also hard to apply to yourself honestly, since the entire problem is that inside and outside feel the same from where you're standing. It works better socially. Ask someone who does know where they think your edge is, and then believe them.

Warren Buffett and Charlie Munger, in the Berkshire Hathaway letters.



18 Sunk Cost

What you've already spent is gone under every option. Decide on what's still ahead.

Nobody disputes the arithmetic. Money and time already spent are identical across every future you can choose, so they can't distinguish between those futures and shouldn't enter the comparison. It survives because it was never an arithmetic error. Abandoning something is a public admission that the spending was a mistake, and what people are protecting is a story about themselves rather than a budget.

In an organisation this compounds, because the story has a name attached. The half-finished migration isn't just an asset with a poor completion cost. It's the thing a director announced last year. Stopping it means somebody is visibly wrong in front of

people who will remember, so the incentive is to keep funding it at a level that avoids the admission without producing the result. Projects like this rarely die. They get starved slowly, which costs more than either finishing or stopping would have.

Any counter-move has to be procedural, because psychology doesn't respond well to being told about itself. Ask the question in a form that removes the history. If this didn't exist and somebody proposed it today, at the remaining cost, with what we know now, would we fund it? A no means the only thing keeping it alive is the spending, and the spending is precisely what shouldn't count.

Separating the two questions helps as well. Was starting this a mistake and should we continue are different questions with different answers, and running them together is what makes the conversation unsurvivable. It's perfectly coherent to say that starting was right given what we knew and continuing is wrong given what we know. Teams that can say that sentence out loud kill bad projects years earlier than teams that can't.

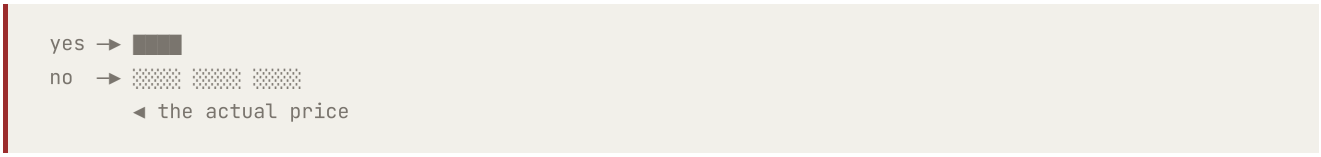
IN PRACTICE

Questions to separate the past from the decision:

- If this did not exist and were proposed today at the remaining cost, would we approve it?
- What is the cost to finish, from here, honestly re-estimated?
- What is the value on completion, re-estimated with what we now know?
- What else could the remaining budget and people do instead?
- Who would have to be visibly wrong for us to stop, and is that shaping the discussion?
- Are we funding this at a level that can succeed, or at a level that avoids a decision?
- What would we need to learn in the next month to justify continuing?

> WHERE IT BREAKS
>
> Applied without judgement it destroys the ability to finish anything. Most projects have a middle stretch where the cost is real and the value is invisible, and a team that re-evaluates from zero every month will abandon work that was three weeks from paying off. The pure form of the rule can't tell that case apart from a genuine trap. Reputation and momentum aren't always irrational either. An organisation that publicly abandons everything it starts pays for it in credibility, and in how willing people are to commit the next time. This is a corrective for one specific bias, not a policy on persistence.

Standard in economics. The behavioural account is Arkes and Blumer, 1985.



19 Opportunity Cost

The cost of what you chose is the best thing you didn't do instead.

The price of a decision is what it displaces, not what it consumes. A quarter spent on one initiative costs you the second-best thing that quarter could have bought, and that thing appears on no document anywhere because it didn't happen. This is close to the most taught idea in economics and close to the least applied in planning, for the straightforward reason that the number you'd need can't be observed.

Roadmaps are where it hides. Every yes is a set of silent noes, and nobody argues for the silent ones, so the comparison never happens. The question in the room is whether this is worth doing, which almost everything passes. The question that would help is whether it's the best available use of the only quarter you have. Reframing a planning meeting from approval to comparison changes what gets built more than any prioritisation framework I've seen.

It applies to attention as sharply as to budget, and attention is the scarcer of the two. A senior person's involvement in one decision is their absence from another. A team's context on one system is context they don't have on the next. Organisations meter money carefully and attention not at all, which is why the expensive mistakes are usually about where the good people were rather than where the money went.

An honest version doesn't need a number. It needs the alternative to have a name. Whenever something is approved, write down what it displaced, and write down the specific other thing rather than a category. That record turns out to be worth more than any estimate, because six months later it's the only way to tell whether the choice was right.

IN PRACTICE

Questions to make the alternative visible before committing:

- What specifically are we not doing because we are doing this?
- Is that alternative better? Who is arguing for it, and if nobody, why not?
- Whose attention does this consume, and what loses that attention?
- Is this the best use of the quarter, or just a good use of it?
- What would have to be true for the second-best option to be the right one?
- If this doubles in cost, does it still beat the alternative?
- What are we permanently giving up by doing this now rather than later?

> WHERE IT BREAKS
>
> Taken seriously it can paralyse you, because there's always a hypothetical better use and imagining one is easier than building anything. It also assumes options are interchangeable and they aren't. A team with deep context in one system doesn't convert cleanly into progress on another, so the alternative on paper is often worth much less in your hands. And some work will never win an explicit comparison and is still worth doing: maintenance, paying down debt, everything that only shows up as the absence of a crisis. Use this to surface the alternative, not to make every choice win a tournament.

Foundational in economics. The standard modern statement is Frédéric Bastiat's essay on the seen and the unseen.

```
they did X
 ├ on purpose ····· rare
 └ nobody noticed ·· usually
```

20 Hanlon's Razor

Never attribute to malice what carelessness explains. It is almost never malice.

When somebody's action harms you, the boring explanation is nearly always the right one. They were busy. They didn't have the context. The message got lost. The ticket had no owner. Nobody thought about your team at all. Malice takes attention and coordination, and organisations are much worse at both of those than they are at neglect.

This is the fundamental attribution error in a form you can use. When other people act, we explain it with their character. When we do the same thing, we explain it with our circumstances. Both explanations are available for both parties and we hand ourselves the generous one every time. The razor is a rule for giving the other side the same terms.

What it changes is what you do next, which is the whole point. If another team broke your flow out of malice, your move is political. If they broke it because nobody ever told them your flow existed, your move is a conversation and a better ownership map. The second assumption is usually correct and it produces a fix. The first produces a faction.

There's a version of it for systems. Before deciding a user is abusing the product, check whether the interface let them believe they were doing the right thing. Most fraud-shaped behaviour in a funnel turns out to be confusion, and treating confusion as fraud builds controls that punish the confused.

IN PRACTICE

Questions before concluding somebody did it on purpose:

- What would have to be true for this to be ordinary carelessness?
- Did they have the context, and did anyone give it to them?
- Would I explain my own version of this with circumstances rather than character?
- Is there an ownership gap here that nobody has ever named?
- What is the cheapest test that separates malice from neglect?
- If it is neglect, what fix stops it happening again?
- Am I building a control for behaviour that is really confusion?

WHERE IT BREAKS

Some behaviour genuinely is malicious or self-interested, and a razor applied indefinitely becomes cover for people who keep doing the same damage. Carelessness that repeats after being pointed out has stopped being carelessness in any useful sense, and at that point the pattern is the evidence. It also fits individuals better than institutions, which can produce sustained harm through incentives with nobody intending anything, and calling that carelessness lets the incentive off the hook. Open with it. Don't end there.

Attributed to Robert J. Hanlon, 1980. The psychology under it is Lee Ross's fundamental attribution error.

VI Borrowed from elsewhere

The four above came out of decision science. These four came out of chemistry, telephone exchanges, systems dynamics and a 1968 paper about committees, and they do more work in product than most things written for product.

```
load 50%  |
     80%  |||
     95%  |||||||||||||
── wait, not throughput ──
```

21 Queueing and Utilisation

Past about eighty percent loaded, waiting time stops rising in a line and starts rising in a wall.

Queueing theory has one result that everybody in product should know and almost nobody does. In a system with variable arrivals and variable service times, average waiting time scales roughly with utilisation divided by one minus utilisation. Half loaded, you wait about as long as one job takes. Ninety percent loaded, you wait nine times that. At ninety-five, nineteen times. The curve is gentle and then it is a cliff, with no warning in between.

So a fully allocated team is not an efficient team. It's a description of a queue that will never clear. Every unplanned request, every bug, every quick question arrives into a system with no slack, and what happens to it has almost nothing to do with how big it is. A two-day task can sit for three weeks and nobody is doing anything wrong.

Utilisation gets optimised because idle time is visible and queue time isn't. You can see an engineer with nothing assigned, and it looks like waste. You can't see the fourteen days a ticket spent waiting, because waiting doesn't appear on anybody's calendar and nobody is billed for it.

Which leads somewhere uncomfortable. Slack isn't waste, it's the mechanism by which things finish on time. A team running at seventy percent delivers faster than the same team running at a hundred. That sounds like a paradox and it's arithmetic.

IN PRACTICE

Questions for when everything is late and nobody is idle:

- What is our real utilisation, counting interrupts, meetings and support?
- How long does a typical item spend waiting, against how long it spends being worked on?
- Which queue is the longest, and can anyone actually see it?
- What arrives unplanned every week, and did we budget for any of it?
- If we left twenty percent deliberately unassigned, what could we then absorb?
- Are we measuring how busy people look or how fast work finishes?
- Where does work sit waiting on one specific person?

> WHERE IT BREAKS
>
> The maths assumes variability. A genuinely predictable pipeline can run near capacity forever, which is why a factory line is not a product team and why the analogy gets abused in both directions. It also says nothing about what should be in the queue: a team at seventy percent building the wrong thing loses to a team at ninety-five building the right one. And it is easy to turn into an argument against ever being busy. The claim is about sustained load, not about a hard fortnight before a launch.

Queueing theory, from A. K. Erlang's work on telephone exchanges. The software framing is Donald Reinertsen's, in The Principles of Product Development Flow.

```
balancing    ▸ change ▸ pushed back
reinforcing  ▸ change ▸ feeds itself
── which one is this? ──
```

22 Feedback Loops

Ask whether the system pushes back against your change or feeds on it. Nothing in between behaves the same way.

This is the most portable idea in systems thinking. A balancing loop resists whatever you do and pulls the system back toward where it was. A reinforcing loop takes any change and makes more of it. A thermostat balances. Compound interest reinforces. Most of what surprises people about organisations comes from mistaking one for the other.

A referral programme reinforces, so it does almost nothing for months and then becomes unreasonable. A support queue balances, because long waits drive people away, which shortens the queue, which is why it feels stable right up until it isn't. Push

hard on a balancing loop and you get very little for the effort, and then somebody concludes the effort was wasted when the loop was just doing its job.

Delay is what makes these hard to see. A loop with a long lag reads as no loop at all. You cut a step from onboarding, the numbers don't move, you decide it didn't matter, and eleven weeks later retention drops for reasons nobody connects back to it. Organisational learning usually fails on lag rather than on measurement.

One property is worth knowing by name. Hysteresis is when reversing the input doesn't reverse the output. You raised prices, churn went up, you put the prices back and churn stayed where it was. The loop moved the system to a new resting place, and undoing the cause does not undo the effect. Assume this every time someone argues a change is safe because it can be rolled back.

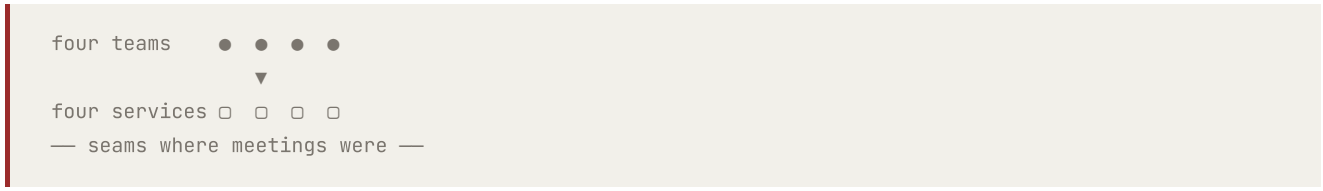
IN PRACTICE

Questions to work out what kind of system you are pushing on:

- Does this system resist what we are doing, or amplify it?
- If it resists, what would have to change for it to stop resisting?
- How long is the delay between the action and the effect?
- Are we about to decide this didn't work before the lag has finished running?
- If we reverse this, does the system come back, or has it settled somewhere new?
- Which loop are we reinforcing without meaning to?
- Who sees the effect first, and are they anywhere near this decision?

> WHERE IT BREAKS
>
> Loop diagrams are easy to draw and hard to check. Anything can be drawn as a loop, and one with eight arrows has all the appearance of rigour and none of the evidence. The model also pushes you to look for a single lever, which real systems seldom have. It explains why an intervention didn't behave as expected, which it does well. It predicts the size of an effect badly, and should not be asked to.

Systems dynamics, Jay Forrester. The readable version is Donella Meadows, Thinking in Systems.



23 Conway's Law

A system copies the communication structure of whoever built it. You ship your org chart.

Conway's 1968 observation was narrow and has aged better than almost anything else written about software. Any organisation that designs a system produces a design whose structure copies that organisation's communication structure. Two teams who talk every day produce a tight interface. Two teams in different buildings produce a queue, a contract, and eventually a service boundary.

For product work the useful direction is backwards. If you know what architecture you want, you have to arrange the people to produce it, because they will produce their own shape regardless of the diagram on the wall. A monolith owned by six teams grows six informal modules whether or not anybody designed them.

It also explains a failure that usually gets blamed on engineering. A flow crossing four teams will feel to the user like it crosses four teams: four inconsistent error messages, four ideas of what a customer is, four places the state can diverge. Nobody chose any of that. It's what the communication structure produced.

The practical use is as a check on any reorg or any architecture proposal. Work out which of the two is being changed, and whether the other one is going to be allowed to follow. Change the architecture without changing the teams and you get the old shape back with new names, usually inside two quarters.

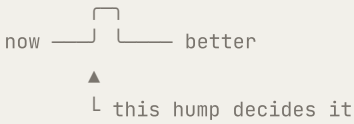
IN PRACTICE

Questions before an architecture decision or a reorg:

- Which team boundaries does this design quietly assume?
- Do those boundaries exist, and if not, who is creating them and when?
- Which parts of this flow cross a team, and does the user feel the seam?
- Where do two teams own the same concept with different definitions?
- If we change the architecture and leave the org alone, what comes back?
- Which interface exists for a technical reason, and which because two people never speak?
- Who has to be in one room for this to come out as one thing?

> WHERE IT BREAKS
>
> It describes a tendency and gets cited as a law. Strong technical leadership routinely produces coherent systems across an incoherent organisation, and a beautifully arranged org still ships badly designed software. Used carelessly it becomes an excuse for any architectural problem and a standing argument for reorganising, which carries costs of its own that nobody counts. Treat it as a prediction about what happens by default when nobody intervenes.

Melvin Conway, How Do Committees Invent?, 1968.



24 Activation Energy

Reactions don't start because the outcome is good. They start when the first step is cheap enough.

In chemistry a reaction can release energy overall and still refuse to happen, because reaching the product means passing through a higher-energy state first. Add a catalyst that lowers the barrier and the same reaction runs at room temperature. The favourable outcome was never what was holding it up.

That maps onto adoption more accurately than most persuasion models do. People aren't failing to use the feature because they doubt it's worth it. They're failing because the first use costs a login, a decision and four minutes they don't have, while the payoff is a week away. Arguing harder about the payoff does nothing to the barrier.

Which is why defaults do more work than any amount of onboarding copy. A default is a catalyst: it removes the initial state change entirely. The most effective product changes are often embarrassingly unglamorous, like pre-filling a field or dropping a confirmation step, and they beat feature work because they act on the constraint rather than on the reward.

The same thing runs inside the team. A code review that takes two clicks happens. One that needs you to find the branch, clone it and build takes a week. Nobody in that story decided reviews weren't important.

IN PRACTICE

Questions for when something obviously worth doing isn't happening:

- What does the first use cost, counted in seconds and decisions?
- How far is the payoff from the effort?
- What could be a default instead of a choice?
- Which single step, removed, would let this start on its own?
- Are we arguing about value when the problem is friction?
- What is the cheapest version of this that still counts as having started?
- Where in our own process is the barrier higher than the work behind it?

WHERE IT BREAKS

Lowering a barrier gets people in and tells you nothing about whether they stay. A frictionless signup in front of a bad product just produces more people leaving. There's an inverse worth respecting too: some barriers are load-bearing, and the confirmation step you removed was the last thing between a user and something irreversible. Work out which kind you're looking at before you take it out.

Chemical kinetics, after Svante Arrhenius. The catalyst half is the part that travels.