

SoftEd

Seeing the Forest for the Trees

Systems Thinking for the AI Era

A practical guide for anyone who is being asked to deploy AI and wants it to improve what customers, clients, and colleagues actually receive.

David Mantica

SoftEd North America

Contents

Introduction: Trees and the Forest	3
Chapter 1: The Local Optimization Trap	5
Chapter 2: Escaping the Trap	8
Chapter 3: Where the Queue Moves in Your Organization	10
Chapter 4: Why Smart People Misread Systems	12
Chapter 5: Causal Loop Mapping	14
Chapter 6: Fixes That Fail, the AI Version	17
Chapter 7: The Iceberg Model	19
Chapter 8: Your Measures Are Part of the Structure	22
Chapter 9: Three Practical Tools	24
Chapter 10: The Forest Check	28
Conclusion: Watch the Forest	30
References	31

INTRODUCTION

Trees and the Forest

AI is exceptional at trees. It cannot see the forest. That one idea runs through this whole book.

A tree is a single task. Draft the memo. Summarize the case file. Answer the support ticket. Write the first cut of the proposal. Write the code. Classify the incoming request.

The forest is everything around the task. Who reviews the memo. Where the case file goes next. What happens to the ticket the chatbot could not answer. Which rules, budgets, and handoffs decide how long your customer waits.

Your organization does not deliver tasks. It delivers outcomes. A customer gets their order. A client gets their answer. A colleague gets the decision they were waiting for. Outcomes live in the forest.

The gains on the tree are real

Start with the good news, because it is real. If you are a skeptic, take these numbers seriously.

Noy and Zhang ran a randomized experiment with 453 college-educated professionals, published in *Science*. The group with ChatGPT finished writing tasks in 40 percent less time. Blind graders rated their work 18 percent higher.

Brynjolfsson, Li, and Raymond studied 5,172 customer support agents at a real company, published in the *Quarterly Journal of Economics*. AI assistance raised issues resolved per hour by about 15 percent on average. The biggest gains went to newer, less experienced staff.

40%

**less time on writing
tasks**

Noy & Zhang, 2023, Science

15%

**more support issues
resolved per hour**

*Brynjolfsson, Li & Raymond,
2025, QJE*

1 task

**the unit both studies
measured**

Neither followed end-to-end flow

What those studies did not measure

Look at what both studies measured. The task. The individual. One person doing one piece of work.

Neither study followed the work after it left that person's desk. That is not a flaw in the research. The researchers asked a task-level question and answered it well.

It is a warning about how you read the research. A 40 percent faster task does not mean a 40 percent faster organization. The gains are real, and they are gains on a tree. Your job is to ask what they do to the forest.

What this book gives you

You are being asked to deploy AI. Most of the advice you get is about the tool. This book is about the system the tool lands in.

- **Chapters 1 to 3** show you the local optimization trap. Speed up one step and the constraint moves.
- **Chapters 4 to 6** teach you to map the feedback and delays that make well-meant fixes fail.
- **Chapters 7 and 8** take you below the waterline, to the structure that produces the problems you keep fixing.
- **Chapters 9 and 10** hand you three tools and one routine. You can start all of them on Monday with no software and no budget.

It does not matter whether you work in a corporation, a nonprofit, a government agency, or a team of ten. Work flows through steps, queues form, and feedback loops push back. The patterns are the same.

Every claim that is not practitioner experience comes from peer-reviewed research. The full list is in the References at the end. The workplace scenarios are illustrative. They are built to look like your world, not to describe any real organization.

The Local Optimization Trap

When you speed up one step of a process and nothing else changes, the system does not get much faster. The constraint moves, and the queue moves with it. That is the local optimization trap.

A process you already know

Take a generic document process. It could be proposals, policy drafts, client letters, or contract language. Forty items arrive every day. Each step can handle a fixed number per day.

Step	Capacity before AI (items/day)	Capacity after AI (items/day)
Intake	40	40
Drafting	20	60
Legal Review	25	25
Approval	30	30
Publish	50	50

Illustrative numbers. Intake is 40 items per day in both cases.

Before AI, the system delivers 20 items a day. Drafting is the slowest step, so the whole system moves at its speed. Twenty items a day pile up in front of the drafters.

Now you run an AI pilot. Drafting capacity triples to 60. The vendor demo was great. The pilot dashboard is green.

Walk it forward. Forty items arrive. Drafting handles all 40 with ease. They move on to Legal Review, which can still clear only 25.

The system now delivers 25 items a day. Not 60. You tripled one step and got a 25 percent gain in what your customers receive.

Look at what you built. Fifteen items a day now stack up in front of Legal Review, every day. Your attorneys did not ask for this. They found out when their inbox doubled.

A system can only move as fast as its slowest step. Speeding up any other step just changes where the waiting happens.

The math behind the pile-up

You might think the example is a cartoon. Real processes are messier. They are, and the math still holds.

Little's Law was published in *Operations Research* in 1961. It is one of the most proven results in the field. It holds as a long-run average for any stable system: factories, help desks, claims departments, software teams.

LITTLE'S LAW

$$\text{WIP} = \text{Throughput} \times \text{Flow Time}$$

$$\text{Flow Time} = \text{WIP} \div \text{Throughput}$$

In plain words: how long you wait depends on how much work is ahead of you and how fast the line moves. Now follow the chain.

1. You speed up a step that is not the constraint. More work reaches the constraint, so work in progress rises.
2. The constraint's capacity did not change. Throughput stays flat.
3. More work in progress divided by the same throughput means flow time rises. The requester waits longer.

A worked example

Fifteen extra items a day stack up at Legal Review. After ten business days the queue holds 150 items. Legal clears 25 a day.

Every new request now waits about six days just to reach a reviewer ($150 \div 25$). The drafting step got faster. The customer got slower service.

That last sentence is the whole trap. Say it out loud the next time someone shows you a task-level AI metric.

Escaping the Trap

The way out is to manage the constraint, not the steps you can speed up most easily. Eli Goldratt's Theory of Constraints gives you five focusing steps to do it. Some of you learned them in a Lean or Six Sigma class. Here they are with AI in mind.

The five focusing steps

- 1. Identify the constraint.** Find where work waits the longest. Not where people complain the loudest. Where work waits.
- 2. Exploit it.** Your scarcest capacity should never sit idle and never do work someone else could do. If your attorneys are formatting documents, that is waste at the constraint.
- 3. Subordinate everything else.** Every other step paces itself to the constraint. Do not flood it just because you can.
- 4. Elevate it.** Only now do you add capacity. People, tools, and yes, AI.
- 5. Repeat.** Once you break a constraint, a new one shows up. Go back to step one. That is not failure. That is how improvement works.

Where AI belongs

Notice where AI sits in that list. Step four. At the constraint.

Not wherever the vendor demo looked most impressive. Not wherever the easiest pilot was. Not wherever a team happened to volunteer.

It is tempting to pick an AI pilot because the task is easy to automate. That is the wrong test. The right test is whether the task sits at the constraint. If it does not, the pilot will look like a success and your customers will feel nothing, or worse.

There is a second lesson in steps two and three. Before you buy anything, you can often get real gains by protecting the constraint's time and pacing the work that feeds it. Those cost nothing.

Does it work?

Mabin and Balderstone reviewed more than 80 published applications of the Theory of Constraints in the *International Journal of Operations & Production Management*. The mean results were large.

Measure	Mean change across published applications
Lead time	70% reduction
Inventory (work in progress)	49% reduction
Throughput	63% increase

Be straight about the caveat, because the people you report to will ask. The authors found no published failures. That tells you people publish their wins.

So treat those numbers as a ceiling, not an average. The direction is still clear. Managing the constraint beats optimizing everything else.

CHAPTER 3

Where the Queue Moves in Your Organization

The new constraint usually lands on a group the AI project never talked to. These three patterns are illustrative, but each one will sound familiar to someone in your organization.

Process	The AI change	The ripple	The new constraint
Software delivery	AI coding assistants help developers write code faster	More code, and larger changes, arrive for review and testing	Code reviewers, QA, and release approval
Proposals and procurement	Your team drafts proposals and RFPs faster; the other side uses AI to answer them	Documents get longer and more numerous	Reviewers, evaluators, and legal sign-off
Customer and IT support	AI resolves routine Tier 1 tickets	What reaches Tier 2 is harder and less familiar	Tier 2 specialists and their capacity

Software delivery

AI coding assistants help developers produce code faster. The pull requests get bigger, and there are more of them.

The constraint moves to code review, testing, and release approval. Your senior engineers, who do most of the reviewing, now spend their days reading code instead of designing systems. Nobody on the AI project was thinking about them.

Proposals and procurement

Your team uses AI to draft proposals and RFPs faster. Vendors and partners use AI to answer them. Documents get longer, and there are more of them.

The constraint moves to whoever has to read, score, and sign off. Those are usually subject matter experts and lawyers who already have full-time jobs.

Customer and IT support

AI deflects password resets and routine questions. That is a real win. But what reaches Tier 2 is now harder, stranger, and less familiar.

The constraint moves to your specialists. Their workload changed in kind, not just in volume. Chapter 6 shows why that matters more than it first appears.

The common thread

In every case, the AI project succeeded on its own terms. In every case, the queue moved somewhere the project plan never looked.

The discipline is simple. Before you deploy, ask one question: **where will the queue form next?** Then go talk to the people who work there.

Sometimes the system does more than move the queue. It pushes back. That is the subject of the next three chapters.

Why Smart People Misread Systems

Capable leaders fall into the local optimization trap because human intuition is built for straight lines, and systems are built from loops and delays. This is not about carelessness. The research shows it happens to very smart people, and it applies to all of us.

Policy resistance

John Sterman at MIT calls it policy resistance. Writing in the *American Journal of Public Health*, he describes how interventions get defeated by the system's own response to them.

The effects that matter show up late and far away. So when they arrive, they look like surprises. They are not surprises. They are the system doing what it was built to do.

If you have ever watched a well-intended rule create a workaround that became a bigger problem than the original one, you have seen policy resistance.

Bathtub dynamics

Booth Sweeney and Sterman tested graduate students at MIT on simple bathtub problems, published in *System Dynamics Review*. Water flows in at one rate and out at another. What happens to the level? Many of these highly educated students could not answer reliably.

Why does that matter to you? Your backlog is a bathtub. Intake is the faucet. Throughput is the drain.

If the faucet opens wider and the drain stays the same, the tub fills. Every queue in your organization is a bathtub. Chapter 1 was a bathtub problem.

Misperceived feedback

In a 1989 study in *Management Science*, Sterman had people run a simple supply chain game: orders, shipments, inventory. They ignored the delays in the pipeline.

They kept ordering because the shipment had not shown up yet. Then everything showed up at once. The result was large, costly swings.

If you have ever over-hired contractors for a backlog, then had nothing for them to do three months later, you have played that game.

The lesson

Flowcharts show steps in a line. They hide delays and loops. That is exactly where the surprises live.

You cannot fix this by trying harder to be smart. You fix it with a tool that makes loops and delays visible. That tool is the causal loop diagram.

Causal Loop Mapping

A causal loop diagram shows how the parts of a system push on each other, including the loops and delays a flowchart hides. You need six symbols to read and draw one.

The six symbols

Symbol	Name	What it means
Aa	Variable	A noun that can rise or fall. "Review queue," not "review." If you can count it, it is a variable.
+	Same direction	A rises, B rises. A falls, B falls.
–	Opposite direction	A rises, B falls. More reviews completed, smaller queue.
R	Reinforcing loop	Change feeds on itself. Drives growth or collapse.
B	Balancing loop	Pushes back toward a goal. Resists change, like a thermostat.
 	Delay	The effect shows up later. The most dangerous mark on the map.

To tell R from B, follow the arrows all the way around a loop. If the loop amplifies whatever change you started with, it is reinforcing. If it pushes back toward a goal, it is balancing.

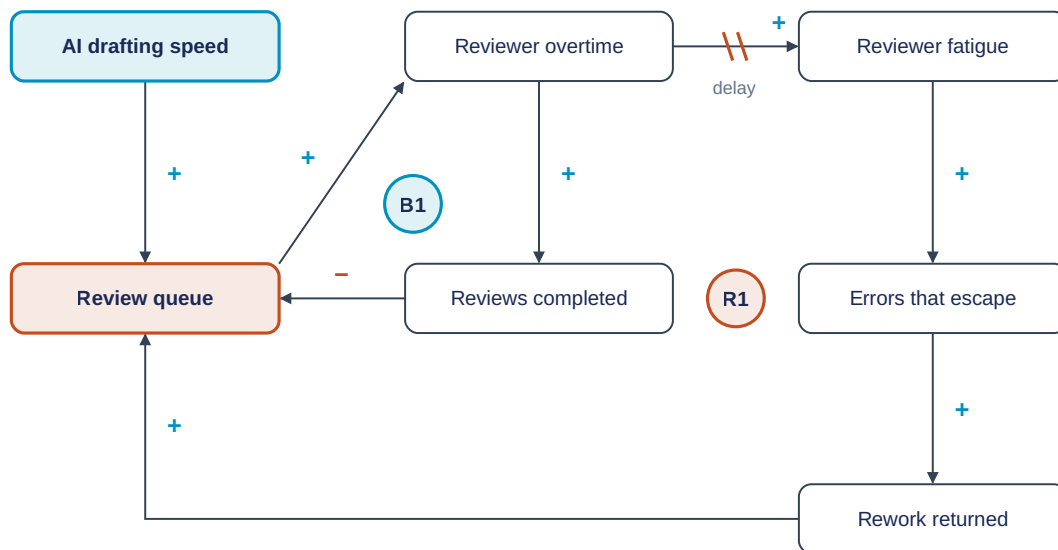
Delays deserve special respect. They are where good decisions go bad. The feedback arrives after you have moved on, reported success, and started the next project.

Three rules keep your maps readable by anyone in your organization. Name variables as quantities. Mark every delay. Label every loop R or B.

Map it: the review bottleneck

Now apply the grammar to the Legal Review problem from Chapter 1.

Overtime clears the queue fast, then refills it slowly



B1 Work harder: overtime clears the queue. Fast relief.

R1 Quality erosion: fatigue, errors, rework. Slow damage.

Follow B1 and the queue falls within weeks. Follow R1, through the delay, and the same queue refills.

Loop B1, work harder. AI drafting speed goes up, so the review queue goes up. When the queue grows, a good manager asks for overtime. More overtime means more reviews completed, and more completions pull the queue down. It works, and it works fast. Within a week or two the queue comes down and everyone breathes.

Loop R1, quality erosion. Follow the other arrow out of overtime, the one with the delay mark. Weeks of overtime build fatigue. Fatigue lets errors escape: the tired reviewer misses the bad citation. Escaped errors come back as rework. The memo gets retracted, the contract amended, the customer case reopened. Rework lands in the same queue. More queue, more overtime, more fatigue, more errors, more rework. That loop amplifies.

This is a documented pattern

This is not a made-up structure. Oliva and Sterman documented it in *Management Science*. Service organizations under pressure work overtime and cut corners, and quality erodes quietly until it becomes a crisis.

Repenning and Sterman describe the same trap in improvement work, in the *California Management Review*. Nobody gets credit for fixing problems that never happened, so people keep working harder instead of working smarter.

What it means for your AI plans

The fast loop tells you the fix is working. The slow loop tells you the truth.

If you watch only the fast loop, you will declare victory right before the damage shows up. The damage arrives as an audit finding, a lost client, or a story you did not want told.

Fixes That Fail, the AI Version

Watch AI deployments for a familiar shape: the symptom improves first and the system degrades second. Systems thinkers call it "fixes that fail." The fix relieves the symptom now. The side effects show up after a delay and often make the original problem worse.

Now: the fix works

Your service desk deploys AI for routine Tier 1 tickets. Password resets, how-do-I questions, known issues.

The queue drops. The dashboard turns green. Leadership declares the pilot a success. Maybe someone gets an award.

Then the delay.

Later: two documented side effects

The ironies of automation. Lisanne Bainbridge published *Ironies of Automation* in *Automatica* in 1983, about industrial control rooms. It is still cited today because the pattern keeps repeating.

When you automate the easy work, people are left with only the hard cases. They also get less practice, because the easy cases were where they built their skills. The job gets harder while the skill base gets thinner.

Think about your newest Tier 2 tech. How did you learn your job? On the easy tickets. Those are gone now.

Automation bias. Parasuraman and Manzey reviewed the research in *Human Factors*. People over-trust automated output. It affects experts, not just novices. Telling people to be careful, or training them, does not make it go away on its own.

Fluent AI text makes this worse. Fluent reads as correct. A clean, confident paragraph does not look like it needs checking, and that is exactly when it does.

So errors escape and come back as rework. That is loop R1 from Chapter 5, arriving by a different road.

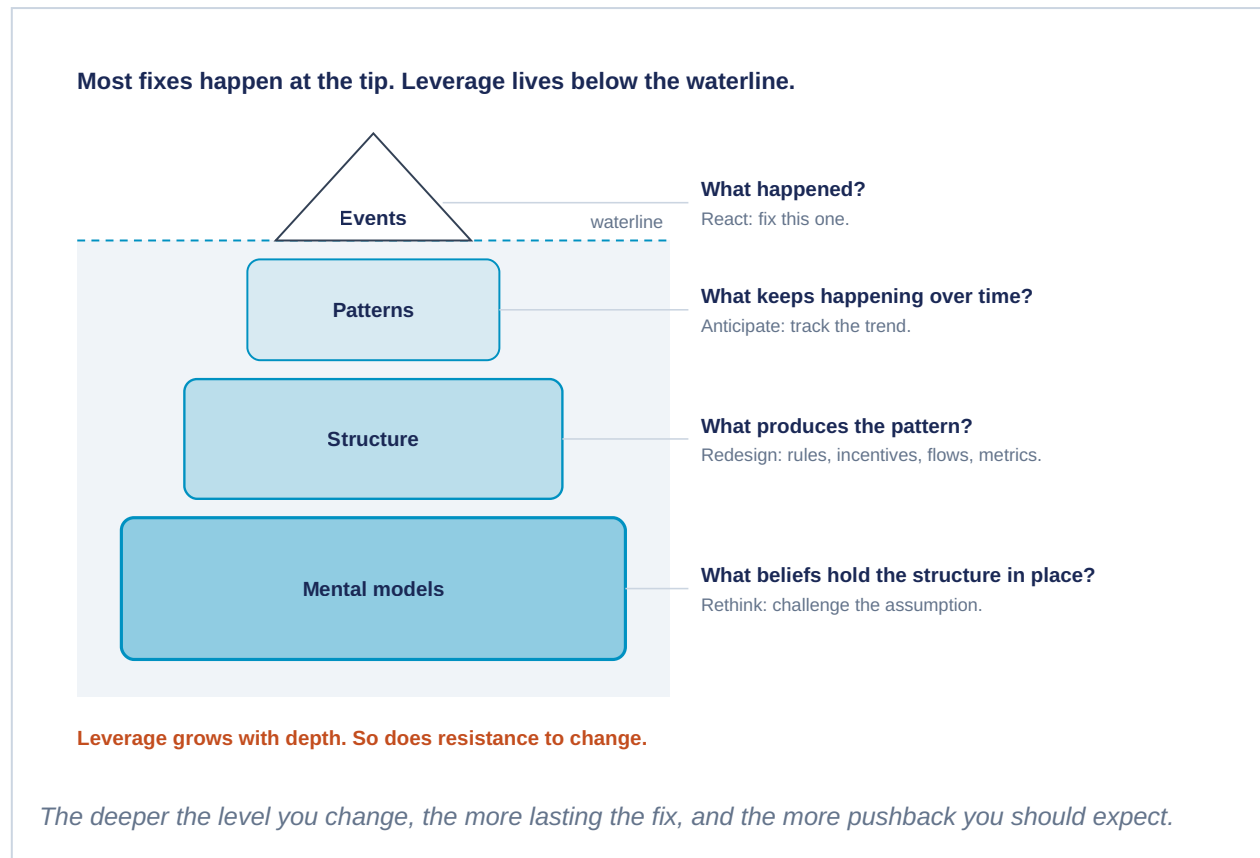
The protective question

Every AI proposal shows you the "now" column. Very few show you the "later" column. You have to ask for it.

One question protects you: what happens after the delay? Ask it in every pilot review, and ask it again 30 days after go-live.

The Iceberg Model

The same problems keep coming back because most fixes happen at the tip of the iceberg, while the structure underneath keeps producing them. The iceberg model gives you four levels to look at, and a different response for each.



The four levels

Events sit above the waterline. The outage. The complaint. The error in the memo. The headline. Events demand a reaction, and most organizations spend most of their energy here. The question is "what happened?" The response is to react and fix this one. That is necessary. Somebody has to put out the fire.

Patterns sit just below the waterline. Is this the first time, or the fifth time this quarter? Is it getting worse? The question is "what keeps happening over time?" The response is to anticipate: track the trend so you see the next one coming.

Structure sits below that. The rules, incentives, workflows, metrics, staffing models, and information flows that produce the pattern. This is what the causal loop map in Chapter 5 shows you. The question is "what produces the pattern?" The response is to redesign.

Mental models sit at the bottom. The beliefs that keep the structure in place. "We've always measured it this way." "Faster is always better." "If the vendor says it's accurate, it's accurate." The question is "what beliefs keep this structure in place?" The response is to rethink.

Leverage grows as you go deeper. So does resistance. That is why most organizations stay at the tip. The tip is safe, and it feels productive.

The model itself comes from systems thinking practice; Peter Senge made it popular. The evidence that structure drives recurring behavior, more than individual effort does, comes from the peer-reviewed system dynamics research in Chapters 4 and 5.

Walk an AI incident down the iceberg

Here is an illustrative scenario. Versions of it have already happened in court filings and published reports, so it is not far-fetched.

An AI-drafted client memo cites a regulation that does not exist. It goes out to 40 clients.

Level	What you find	Your response
Event	The memo is retracted and corrected. An apology goes out.	React: fix the memo.
Pattern	Review minutes per AI draft have fallen for three months while draft volume doubled.	Anticipate: chart review time against volume.
Structure	The unit is scored on drafts published. No reviewer time was budgeted when AI arrived. The review checklist predates AI and never says "verify every citation."	Redesign: budget review time, update the checklist, add a citation check.
Mental model	"If it reads fluently, it's probably right." That is automation bias (Parasuraman & Manzey).	Rethink: fluent is not verified.

If you stop at the event, you will be doing this again. Nobody was watching the pattern, so the incident looked like a surprise. It was predictable.

The structure produced the pattern, and the pattern produced the event. Each level has its own response. If you only fix the memo, you will fix the next memo in six weeks, and the one after that.

The mental model is where your words matter most. Fluent is not verified. That sentence belongs in every AI policy and every AI training session in your organization.

Your Measures Are Part of the Structure

Measure AI by end-to-end outcomes, not by the speed of the step it touched, because whatever you measure becomes part of the structure that drives behavior. Of all the structural levers in Chapter 7, measures are the one you control most directly.

What the research found

Gwyn Bevan and Christopher Hood studied performance targets in the English public health system, published in *Public Administration*. The setting was public health care, but the pattern shows up in any organization that runs on targets. They documented three forms of gaming that show up again and again.

Effect	What happens	What it looks like with AI
Ratchet effect	Next year's target is set from this year's result, so units hold back.	Teams quietly underreport AI time savings. Report 30 percent savings and someone may cut 30 percent of the budget or add 30 percent more work.
Threshold effect	Performance converges on the target line. Strong performers coast down to it.	Policy says every AI draft gets one human review, so every draft gets exactly one, no matter how risky. The minimum becomes the standard.
Output distortion	The measured number rises while unmeasured quality falls.	Drafts per week soar. Accuracy is not counted, so it slips. Nobody sees it until the retraction.

None of this needs bad people

Bevan and Hood were not describing villains. They were describing what reasonable people do inside a measurement system. That is structure at work.

So do not respond to gaming with a speech about integrity. Respond by changing what you measure.

Better measures for AI

Measure what your customers actually receive and how long they wait for it. Some practical shifts:

- **Measure flow time end to end**, from request to delivery. Not the time of the step the AI touched.
- **Pair every speed measure with a quality measure**. Drafts per week travels with errors found after release.
- **Track the queue in front of the constraint**. If it grows after an AI deployment, the deployment moved the constraint, whatever the pilot report says.
- **Count rework**. Retractions, amendments, appeals, and reopened tickets are the R1 loop made visible.

If a metric could go up while your customers' experience gets worse, it is not a good AI metric on its own.

Three Practical Tools

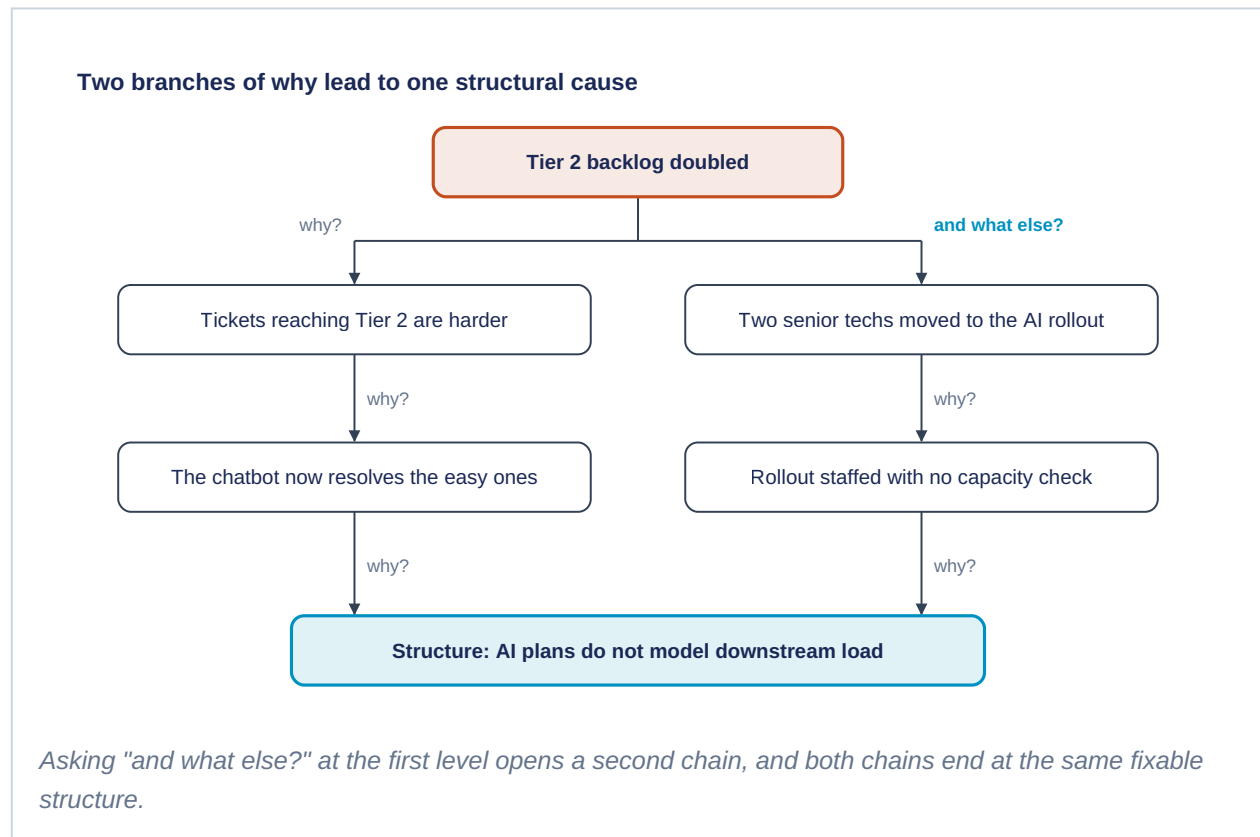
Three habits turn everything in this book into practice: five whys pointed upstream, constraint tracking, and leverage point questions. None of them need software, a consultant, or a budget line. You can start all three on Monday.

Tool 1: Five whys, pointed upstream

Most of you know the five whys from the Toyota Production System. Something goes wrong, you ask why, and you keep asking until you reach the root cause.

Alan Card reviewed the technique in *BMJ Quality & Safety* and found a real weakness. Classic five whys follows one chain to one root cause. Real failures in complex systems have several causes, and the number five is arbitrary.

You do not need to drop the technique. Change two things: let it branch, and stop at structure.



Tier 2 backlog doubled. Why? The tickets reaching Tier 2 are harder. Classic five whys would keep going straight down that chain. Instead, ask the second question: **and what else?** Two senior techs were pulled onto the AI rollout. That cause is just as real.

Follow both branches. The tickets are harder because the chatbot now resolves the easy ones. The techs were pulled because the rollout was staffed with no capacity check. Both branches land on the same structure: your AI deployment plans do not model downstream load. That is a planning template change, not a performance conversation.

Four rules make it work:

- 1. Point it upstream.** At process and structure, never at a person. The moment a why lands on a name, you stop learning and start blaming, and people stop telling you the truth.
- 2. Branch, do not chain.** At every level, ask "and what else?"
- 3. Stop at structure.** Stop when you reach a rule, a flow, a metric, or an incentive you can change. Not when you reach the number five.
- 4. Check with data.** Every answer should be something you can observe. "The tickets are harder" should come with ticket data.

The upstream habit is the real tool. When something breaks, your first question stops being "who?" and becomes "where did this come from?"

Tool 2: Constraint tracking

Constraint tracking turns the Theory of Constraints from Chapter 2 into a weekly habit. Ask four questions every week:

- 1.** Where is work waiting the longest right now?
- 2.** How much work is sitting in front of it, and is that growing?
- 3.** Did anything we changed this week move the queue?
- 4.** Is everything else paced to the constraint, or are we feeding it faster than it can clear?

Log the answers. Here is what six weeks can look like (illustrative numbers):

Week	Constraint	Queue (items)	Wait (days)	What changed
6	Legal review	55	6	Better review checklist; paced release into review
3	Legal review	110	12	Constraint moved after the AI pilot
1	Drafting	60	9	AI drafting pilot starts

In week 1 the pilot looked like a win. By week 3 customers were waiting three days longer, and a report on drafting speed alone would have called it a success. By week 6 the team had protected the constraint (Goldratt's step two) and paced the work feeding it (step three), and the wait fell below where it started.

You do not need a system for this. A shared spreadsheet and 15 minutes in your weekly staff meeting are enough. One rule: **re-find the constraint after every AI change. It will move.**

Tool 3: Leverage point questions

Donella Meadows identified twelve places to intervene in a system, ranked from weak to powerful. Abson and colleagues, writing in *Ambio*, grouped her twelve into four realms. That makes them usable in a meeting.

Realm (shallow to deep)	Examples	The question to ask
Parameters	Headcount, budgets, thresholds, AI licenses	What number are we changing, and will the system simply absorb it?
Feedbacks	Delays, reporting cycles, who hears about errors	Who learns what, and how fast? Can we shorten the delay?
Design	Rules, information flows, decision rights	Who sees what, and who decides? What rule sends work into the queue?
Intent	Goals, the mission, the underlying paradigm	What is this system actually for? Is AI serving that purpose, or serving a metric?

Parameters are easy to change, but the system often absorbs the change and returns to what it was doing. Feedbacks change behavior when you shorten them: if an AI-drafted error goes out, how long until the drafter hears about it? Design decides who can see the review queue and who can stop a release.

Intent is the deepest. A support desk exists to solve customers' problems, not to close tickets as fast as possible.

Many AI initiatives stop at parameters. Buy the tool, add capacity to one step, declare victory. Before you sign off on your next AI project, ask the question in every row. The deeper questions are where your real leverage is.

The Forest Check

Run the Forest Check before every AI deployment and again 30 days after go-live. It pulls every tool in this book into one routine of four steps. A whiteboard and an hour with the right people is enough.

01 Map the flow Where does work wait today? TOOL Constraint tracking	02 Predict the move If this step gets faster, where does the queue form next? TOOL Five focusing steps and Little's Law	03 Draw the loops What feedback and delays will this change trigger? TOOL A causal loop map	04 Go deep Are you changing an event, a pattern, the structure, or the intent? TOOL Iceberg, upstream whys, leverage questions
--	---	---	--

The second check, 30 days after go-live, gives the delays from Chapter 5 time to surface. It catches what the first check missed.

Before you deploy

Copy this checklist into your project plan.

- ☐ We know where work waits longest in this process today, and we have the numbers.
- ☐ We know whether the step AI will speed up is the constraint.
- ☐ We have predicted where the queue will form next, and we have talked to the people who work there.
- ☐ We have protected the constraint's time and paced the work feeding it before adding capacity.
- ☐ We have sketched the loops this change could trigger, with every delay marked.

- ☐ We have asked what happens after the delay, and written down what we will watch.
- ☐ Our success measure is end-to-end flow time and quality, not the speed of one step.
- ☐ Every speed measure is paired with a quality measure.
- ☐ We have asked the leverage questions for parameters, feedbacks, design, and intent.
- ☐ Reviewers know that fluent is not verified, and review time is budgeted.

Thirty days after go-live

- ☐ We have re-found the constraint. We know where it moved.
- ☐ We have checked the queue in front of the new constraint, and whether it is growing.
- ☐ We have compared end-to-end flow time to where it was before the deployment.
- ☐ We have counted rework: retractions, amendments, appeals, and reopened tickets.
- ☐ We have walked any incident down the iceberg, not just fixed the event.

CONCLUSION

Watch the Forest

AI will keep getting better at trees, so your value is in watching the forest. Every year the tools will be faster and more fluent. That is not in question.

What is in question is whether the speed reaches your customers. That depends on the queues, loops, measures, and beliefs around the task. Those belong to you, not to the tool.

Five things to take back to your team

- 1. AI optimizes the tree. Your job is the forest.**
- 2. Speed up a step that is not the constraint and you get a longer queue, not more output.** Your customer waits longer.
- 3. Map the loops.** Flowcharts hide the delays, and the delays are where fixes fail.
- 4. Events are symptoms. Structure produces them.** Your measures are part of that structure.
- 5. Use the tools every time.** Upstream whys, constraint tracking, and leverage questions. Run the Forest Check before every AI deployment and 30 days after.

The people who keep an organization running are often invisible when things work. The order ships. The invoice is right. The system stays up at two in the morning.

Make sure someone in the room is watching the forest. After reading this, that can be you.

References

Every research claim in this book comes from these peer-reviewed sources.

- Abson, D.J., et al. (2017). Leverage points for sustainability transformation. *Ambio*, 46(1), 30-39.
- Bainbridge, L. (1983). Ironies of automation. *Automatica*, 19(6), 775-779.
- Bevan, G., & Hood, C. (2006). What's measured is what matters: Targets and gaming in the English public health care system. *Public Administration*, 84(3), 517-538.
- Booth Sweeney, L., & Sterman, J.D. (2000). Bathtub dynamics: Initial results of a systems thinking inventory. *System Dynamics Review*, 16(4), 249-286.
- Brynjolfsson, E., Li, D., & Raymond, L. (2025). Generative AI at work. *Quarterly Journal of Economics*, 140(2), 889-942.
- Card, A.J. (2017). The problem with "5 whys." *BMJ Quality & Safety*, 26(8), 671-677.
- Little, J.D.C. (1961). A proof for the queuing formula $L = \lambda W$. *Operations Research*, 9(3), 383-387.
- Mabin, V.J., & Balderstone, S.J. (2003). The performance of the theory of constraints methodology. *International Journal of Operations & Production Management*, 23(6), 568-595.
- Noy, S., & Zhang, W. (2023). Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654), 187-192.
- Oliva, R., & Sterman, J.D. (2001). Cutting corners and working overtime: Quality erosion in the service industry. *Management Science*, 47(7), 894-914.
- Parasuraman, R., & Manzey, D.H. (2010). Complacency and bias in human use of automation: An attentional integration. *Human Factors*, 52(3), 381-410.
- Repenning, N.P., & Sterman, J.D. (2001). Nobody ever gets credit for fixing problems that never happened. *California Management Review*, 43(4), 64-88.
- Sterman, J.D. (1989). Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. *Management Science*, 35(3), 321-339.
- Sterman, J.D. (2006). Learning from evidence in a complex world. *American Journal of Public Health*, 96(3), 505-514.

Two models in this book come from practitioner books rather than journals: the iceberg model (popularized by Peter Senge in *The Fifth Discipline*, 1990) and the five focusing steps (Eli Goldratt). Donella Meadows' original list of leverage points is also a practitioner essay (1999). The claims made about all three rest on the peer-reviewed sources above.

SoftEd

*AI will keep getting better at trees.
Make sure someone in the room is watching
the forest.*



Connect with David Mantica on LinkedIn

softed.com