After the Prompt

The Operating Model for Companies
That Want More Than AI Productivity

Jean-Philippe LeBlanc

jpleblanc.com
aftertheprompt.co

Contents

Product and Strategy

The roadmap as an intent system

Chapter 10

The Intent-Driven Organization

What it looks like when a company has actually written down what it wants

Chapter 11

The Leader's New Job

From approving output to designing the conditions for it

Chapter 12

The 90-Day Transition

A practical playbook, and what to do when it goes sideways

The Operating Model for Companies That Want More Than AI Productivity

Jean-Philippe LeBlanc

0

PREFACE

Before We Begin

A note on the cases, and on the author

A note on the cases in this book

The work in these pages came out of years of conversations with leaders trying to put AI into organizations that were not designed for it. Some happened on the record. Most did not. The book owes a debt to people who would rather not see their company name attached to a candid quote about their own org chart.

I am not a consultant. I did not run engagements with the teams in these pages. The cases come from conversations: hallway exchanges at conferences, long phone calls with people I met through my CircleCI years, dinners where someone three glasses in was finally willing to say what was actually happening inside their function. Where a person is named in full, that is their real name, and they have signed off on the quote. Where a person is named by first name only, I have changed the name. The role, the company size, the industry, and the substance of what they said are accurate. Where a company appears generically (a mid-stage fintech, a 2,000-person insurance company, a publisher), I have anonymized on purpose.

The numbers are real where the leader told them to me. The six-hour-to-ninety-minute review cycle is what one general counsel told me her team measured. The fifty deploys a day at Etsy in 2013 is

published. Some timelines are compressed for readability. A conversation that ran over multiple exchanges may appear as a single one.

The argument is not the anecdotes. It stands or falls on whether the pattern holds, and the pattern is what I am asking you to test against your own organization.

Author's note

I did not come to this book through AI. I came to it through pipelines.

For most of my five years at CircleCI, I sat with engineering teams while they argued about what "done" meant. Not in the abstract. In the literal sense of writing it down so a build system could decide whether to ship code to production without asking anyone first. That work, repeated across thousands of teams, shaped how I think about everything else.

The pipeline forced a kind of honesty most organizations never have to face. You could not run a deployment on good intentions. You could not run it on a slide. You had to declare, in a file the machine could read, exactly what you wanted to be true before you let the next step happen. The teams that did this well moved faster than seemed possible. The teams that did it poorly suffered in a way the others had forgotten existed.

When the current wave of AI tools landed in non-engineering parts of the business, I watched leaders make the same mistakes engineering had made in the years after 2009 and then solved. They treated AI as a productivity layer. They prompted their way to faster outputs. They built dashboards showing time saved. They ran into the same wall, except they did not have the vocabulary to name it.

This book is not about better prompting. There are plenty of those books, and most are fine. This book is about what comes after: the operating model that lets an organization act on its own intent at machine speed without burning a human to translate every request.

I have been wrong about pieces of this argument. I will name where, as we go. The strongest part of my perspective is not that I figured anything out alone. It is that I watched a lot of teams figure things out, a lot fail, and the difference was rarely what the teams themselves thought it was.

1

CHAPTER 1

The Ceiling You Can't See From the Floor

Why success at prompting is not success at transformation

You got the win.

The pilot shipped. The CFO signed off on the business case. Your team built dashboards showing time saved, tickets deflected, drafts produced faster. Somebody in the C-suite said your name in a town hall. The deployment was real. The numbers were real.

And still, on a Thursday evening after the rollout dust settled, you sat with a feeling you could not quite name. Not regret. Something quieter. A sense that the thing you built works, and the thing you wanted to change did not change.

You probably have not said this out loud. Saying it sounds like complaining about your own success. So you keep it to yourself, and you keep optimizing.

I have watched this happen with enough leaders to call it a pattern, not a personality flaw.

The quiet dissatisfaction

You run a function. Marketing, operations, product, legal, finance, some slice of a company that matters. You pushed for the AI investment. You did the vendor evaluation or the internal build. You sat

in the rooms where people were skeptical, and you made the case with data and patience and a few well-timed demos.

It worked. People adopted the tools. Usage climbed. Reports got written faster. Customer responses got drafted in half the time. Someone on your team figured out a prompt that turned a three-day analysis into a forty-minute one, and for a week you felt like you were standing on the edge of something real.

Then the forty-minute analysis became normal. The next quarter looked a lot like the last, except a little faster.

The dissatisfaction is hard to pin down because nothing went wrong. The ROI is positive. The spreadsheet says you won.

The work feels the same. The meetings are the same meetings. The decisions get made the same way. The org chart has not moved. The questions your team asks are the same questions, answered quicker. You have a private suspicion that speed was never the thing that needed to change.

The feeling is accurate. It is the correct response to having done everything right inside a container that was always going to be too small. The leaders I spoke with in 2024, 2025, and 2026 were not undercelebrating. They were registering, accurately, the difference between speed and difference. Their tools got faster. Their organizations did not become anything new.

That gap is what this book is about.

What the metrics measured and what they did not

The business case you built, and I have read dozens, almost certainly measured time. Time to draft. Time to summarize. Time to respond. Time to first version. Sometimes volume. Occasionally error rates or rework cycles.

Real gains. I am not dismissing them.

A general counsel I spoke with in late 2025 told me her team had cut contract review prep from six hours to ninety minutes. She ran legal for a 2,000-person insurance company. People went home earlier. The backlog shrank.

What nobody measured was whether the legal team started making different decisions about which contracts to pursue. Whether faster review changed how the company thought about risk. Whether anyone upstream adjusted their assumptions about what legal could take on now that capacity had

shifted.

The business case captured the task. It did not capture the decision the task feeds into. It captured the speed of the answer, not whether the question changed.

An organization that does the same things forty percent faster is still the same organization.

I have sat through versions of this slide across most of the large organizations I have spent time with. The shape is consistent: a speed metric up, the outcome metric on the same work never instrumented. Fifty-two percent reduction in claim-processing turnaround time. Four people clap. Nobody has asked what happened to the denial rate on those claims. The same slide runs next quarter with a bigger number.

Back to the general counsel I will call Priya. She had pushed for the deployment because she thought faster review would let her team get upstream of deals. She wanted to advise on structure before contracts were drafted rather than reviewing them after. That did not happen. The deal teams kept sending finished contracts. The faster turnaround just meant legal said yes or no sooner.

Priya said it like this, on a Zoom that ran long because she was tired and willing to be honest: "We became the fastest version of what we already were."

I have thought about that line ever since. It is the cleanest articulation of the problem I have heard from any leader in any function.

The prompt is a tax

Every time someone sits down to write a prompt, something specific is happening that looks like productivity and is actually a cost.

They stop what they are doing. They translate what they need from the messy language of their problem into the structured language the model responds to. They review the output. They adjust. They re-prompt. They copy the result into wherever it needs to go. Then they do it again for the next thing.

Every cycle of value requires a human to stop, translate, and interpret. Every one.

Gloria Mark's research at UC Irvine found it takes an average of twenty-three minutes and fifteen seconds to return to full focus on a task after an interruption. A prompt cycle is a small interruption. The attention cost compounds across a team, across a day, across a quarter.

A product manager at a B2B SaaS company, a guy I will call Derek who was gifted at this work, built a seven-step prompt chain that turned raw customer feedback into prioritized feature recommendations. Twenty minutes a week. Down from two days.

Months later, Derek was still running it. Manually. He was still the bottleneck. If Derek went on vacation, nobody ran it. If Derek left, the chain would rot within a month because nobody else understood the assumptions baked into each step.

The value was real. It was trapped inside Derek's weekly attention.

The prompt is not a lever. It is a tax. A recurring one. And the tax rate does not go down.

I keep hearing about teams hitting a point where the time saved by AI gets consumed by the time spent operating AI. The treadmill comes with better shoes.

Implicit intent at machine speed

The reason your gains stayed trapped at the task layer is not that the model is too small or the prompts not clever enough. It is that the thing the AI is acting on, your organization's actual intent about what matters and in what order and against which trade-offs, was never written down. It lives in your head, in Priya's head, in the head of the VP of Marketing who said "focus on mid-market" in a quarterly planning meeting and now wonders why five teams interpreted that differently.

Implicit intent is what every organization has always run on. Strategy in conversations. Priorities in slides. Trade-offs adjudicated in one-on-ones by experienced managers who could read the room and split the difference. It worked because humans were the only actors in the system, and humans translate lossily but correctively. Your manager says one thing, you do a slightly different thing, she sees the result, she corrects, you converge.

This is not a flaw. It is a quiet feat. The amount of organizational coordination that runs on shared context, accumulated judgment, and the ability to ask "wait, what did you mean by that" is enormous.

The moment you add a non-human actor, it breaks. Not gradually. Immediately.

The AI cannot read the room. It cannot infer from your tone that you meant something slightly different than you said. It does what you literally asked, which is, every time, a subtly worse version of what you actually wanted. So you re-prompt. You correct. You explain. The correction loop you used to run between humans is now a correction loop you run between yourself and a machine that does not learn between sessions.

A junior on your team gets corrected once, internalizes the pattern, and starts applying it. The model does not. Every cycle is the first cycle. Your accumulated organizational judgment, the part of leadership that used to compound, has nowhere to live in the system.

This is the villain of the book. Implicit intent at machine speed. The cost of organizational ambiguity, which was always there, was always expensive, and was always absorbed silently by humans rewriting briefs and managers correcting drift, becomes visible and immediate the moment a system acts on the ambiguity faster than your correction loop can catch up.

The AI did not create this problem. The AI revealed it.

That is a different sentence than the one in the press releases. It is also the only one that explains what you are feeling.

Why now

The first wave of enterprise AI was adoption. Pick a vendor, train a team, count the seats, log the time saved. Most companies are still finishing this wave.

The second wave is orchestration. Multiple tools, chained calls, agents that hand work to other agents, retrieval pulling from systems the original procurement decision never anticipated. The interesting work has already moved here. The interesting failures too.

The third wave will be organizational intent. As tools become agents, workflows, copilots, and ambient systems that act on a company's behalf without a human in the seat each time, the thing being executed is no longer a prompt. It is the company's policy. Its priorities. Its trade-offs. Its definition of "good." Most of which has never been written down in a form a machine can read.

A bad prompt creates a bad output. Bad organizational intent creates bad action at scale.

The companies that do not make intent explicit will do more than use AI badly. They will automate their own confusion. The lossy oral tradition that used to be corrected by a manager in a one-on-one will be executed by a system that does not pause for the one-on-one. The drift that used to surface in a quarterly review will surface in production, after the customer has already received the email or the denial.

The window is shorter than most boards think. The companies setting up agent infrastructure right now are about to learn whether their organization knows what it actually wants. The ones that do will accelerate. The ones that do not will accelerate in three different directions at once.

A ceiling is not a failure

The prompt-and-response model was the right first move. It had to be first, because it was the only thing that was ready. The models needed a human in the loop. The organizations needed to learn what AI could do by doing it, case by case, prompt by prompt. The failure modes had to be discovered by real people in real workflows making real mistakes.

You did that work. The thing you built is not broken. It is complete.

There is a difference.

A ceiling is not a crack in the foundation. It is the top of a room you have fully occupied. You put up the shelves, hung the pictures, made it work. Now you are standing in it, looking up, wondering why it does not feel like enough.

For about eighteen months I gave leaders the wrong advice when they asked. Push harder. Prompt better. Find more use cases. Every word of that pointed deeper into the same room. What they needed was a different relationship between the organization and the machine, one where the human is not always the translator, where the system does not wait to be asked, where the value is not extracted one query at a time.

Prompting helps people work faster. Intent architecture helps organizations work differently.

That is the sentence the rest of this book is going to defend.

One part of every company has already lived through a version of this. Most leaders outside that function still believe the lesson does not apply to them.

The Prompt Ceiling Diagnostic

You can run this in ten minutes, alone, with a notebook. Do not workshop it. Do not turn it into a survey. The point is whether you, the leader who pushed for the AI investment, can answer these honestly to yourself before answering them to anyone else.

- Are we measuring time saved more than decisions changed?
- Are humans still manually translating every task into AI instructions?
- Do workflows stop when the expert is unavailable?
- Are we producing more output without changing prioritization?

– Are teams saying the AI was right on the brief and wrong on the point?

Three or more yeses means you are not failing. You are at the ceiling. The actions that got you here will not get you out. The next chapter is about a part of your company that has been past this ceiling for a decade and has, in most organizations, never been asked what it learned.

2

CHAPTER 2

The Pipeline Was Always the Point

What engineering already learned about machine-readable intent

In 2013, a software team at Etsy was deploying code to production fifty times a day.

That number meant almost nothing to people outside engineering. Inside, it signaled that something fundamental had changed about how humans and systems worked together. Not the tools. Not the speed, exactly. The relationship.

Ten years earlier, the standard practice at most software companies looked like this. A developer finished a piece of code, handed it to someone else, and that someone else ran a manual checklist to get it into production. Check the dependencies. Run the tests. Stage it. Verify. Push it live. Hope.

Each step required a person to decide whether to proceed. Each handoff was a chance for miscommunication. Each decision point was a moment where someone's judgment, mood, or interpretation of "ready" could send the whole thing sideways. When something broke, the first question was always who touched it last.

The problem was structural. The system depended on humans translating intent at every step, and humans translate differently depending on the day.

How engineering stopped asking every time

The fix, when it came, was not a better checklist.

It was a different kind of object. Engineering organizations started writing down what "done" meant, precisely and completely, in configuration files a machine could read. Not "run the tests" but which tests, in what order, with what thresholds for passing, and what happens if they fail. Not "deploy to staging" but which staging environment, with which version of which dependencies, verified against which conditions before the next step runs.

This is what a CI/CD pipeline is. The idea underneath the acronym is plain. You write down your intent once, with enough precision that the system can execute it without asking you again.

At CircleCI, I watched thousands of engineering teams go through a version of this transition. The tooling was never the interesting part. The interesting part was the moment a team realized that writing the pipeline forced them to answer questions they had been avoiding.

What counts as a passing test? Not in theory. In practice. Right now. For this service. A test that takes too long, is that a failure or a warning? A dependency one patch version behind, does that block the deploy or just log a note? Who decides the order of operations, and what does that order reveal about what the team actually values?

An engineer I know at a mid-stage fintech once described a three-day argument his team had over whether linting errors should block deployment. Three days. Engineers walked out of meetings. One senior engineer threatened to escalate to the CTO. The argument was not about linting. It was about what the team believed "quality" meant and whether they trusted each other to fix things after the fact. The pipeline forced that argument into the open because the pipeline needed an answer. A human could shrug and say "it depends." The configuration file could not.

That is the shift. Not from manual to automated. From implicit to explicit. From "we all kind of know what good looks like" to "here is what good looks like, written down, and the machine enforces it."

Once the pipeline existed, something nobody planned for happened. The pipeline became a record of the team's decisions. You could look at a configuration and see what the team cared about, what they were willing to tolerate, where they had made trade-offs. Organizational intent, made legible.

The second-order property, the pipeline as a visible record, turned out to be the whole point. The first-order property, faster deploys, was the byproduct.

Intent is not strategy held in someone's head

There is a difference between knowing what you want and writing it down precisely enough that someone else, or something else, can act on it without calling you first.

Most leaders I talk to have strong strategic intent. They know what their function is supposed to accomplish. They can articulate priorities in a room. Ask a VP of Marketing what matters this quarter and she will give you a clear answer: "Pipeline generation in mid-market, with a focus on two specific verticals." She means it.

Now ask her three direct reports, separately, what matters this quarter.

You will get three different answers. Not wildly different. Recognizably related. But different enough that the work those three people approve, prioritize, and resource will drift apart over time. Not because anyone misunderstood. Because the translation from her head to theirs was lossy. It always is.

Explicit intent is different. It is written down with enough specificity that a system can act on it. "Pipeline generation in mid-market, with a focus on two specific verticals" is a direction. "Every piece of content we produce between now and June targets companies with 200 to 2,000 employees in healthcare or financial services, prioritizes demo requests over whitepaper downloads, and is measured on influenced pipeline, not MQLs" is something a system can actually work with.

The gap between those two statements is the gap between strategy and execution in most organizations. It is the same gap engineering teams closed when they started writing pipelines.

The problem is not the thinking. The problem is the medium. Conversation is lossy. Slides are lossy. The human chain of interpretation from executive to manager to individual contributor to system is lossy at every link.

Engineering figured this out because engineering had no choice. The machines would not run on vibes.

The oral tradition of organizational direction

For decades, the lossiness did not matter much.

Organizational knowledge about what mattered, where to focus, which trade-offs were acceptable, all of it lived in people's heads and got passed around through meetings, hallway conversations, Slack threads, and the accumulated judgment of experienced managers. The oral tradition is not a bug that needed fixing. It was an operating model that succeeded because humans were the only actors in the system.

If every decision-maker, executor, reviewer is a person, the lossiness of human communication gets corrected by human judgment. Your manager says "focus on mid-market." You interpret it slightly differently than she meant. She sees the work you produce. She corrects. You adjust. The signal gets refined through iteration, and most of the time the system converges close enough to the right answer.

This is how most organizations still work. Not just small ones. Big public companies with strategy decks and OKRs and quarterly business reviews. The formal artifacts of strategy exist, but the actual direction still lives in the interpretive layer between the artifact and the person reading it.

The correction loop was fast enough and cheap enough. A manager could catch a misalignment in a weekly check-in and redirect. The cost was time, attention, and a lot of meetings. Manageable.

Until it was not.

The moment you introduce a non-human actor into the system, one that cannot ask clarifying questions, cannot read the room, cannot infer what you probably meant from the expression on your face, the oral tradition breaks. Most organizations are in this moment right now and do not recognize it clearly. They have systems that need to act on organizational intent, and organizational intent was never written down in a form those systems can read.

Imperative versus declarative

In programming, there is a distinction between telling a system what to do step by step, and telling a system what you want to be true and letting it figure out how.

The first is imperative. "Go to the database. Find all the records from January. Sort them by date. Remove the duplicates. Send the result to the dashboard." You specify every step, in order.

The second is declarative. "The dashboard should show unique records from January, sorted by date." You specify the outcome. The system handles the steps.

A technical detail on its face. The most important organizational idea in this book.

Priya's legal team, cutting contract review from six hours to ninety minutes, was operating imperatively. Someone sat down, opened the contract, prompted the AI with specific questions in a specific sequence, reviewed the output, corrected it, and passed it on. Do this. Then do this. Then do this.

What Priya actually wanted was declarative. She wanted a state of the world where contracts meeting certain criteria got flagged automatically, where risk thresholds were applied without someone

remembering to check, where the output of the review fed directly into the deal team's decision process. She did not want faster steps. She wanted a described outcome the system maintained.

The difference is not about automation. Automation just speeds up the imperative sequence. The difference is about who holds the logic. In an imperative system, the human holds it and feeds it to the machine one instruction at a time. In a declarative system, the logic is written into the system and the human describes the desired state.

A sales team that runs on "call the leads your manager tells you to call" is imperative. A sales team that runs on "every inbound lead scoring above 75 gets a call within four hours, prioritized by deal size" is declarative. The second can be acted on by a system. The first requires a manager's attention every time.

The organizations stuck at the ceiling from the last chapter are almost always operating imperatively. They are telling the AI what to do each time, rather than declaring what they want to be true.

The cost of ambiguity has always been there

Here is the thing I got wrong for the longest time.

I assumed the cost of strategic vagueness was new. That it emerged when AI showed up. That before AI, ambiguity was fine.

That is backwards. The cost was always there. It just did not have a line item.

Every team that built the wrong thing because the brief was ambiguous was a cost. Every pair of departments that duplicated effort because nobody had written down who owned which part of the customer experience was a cost. Every new hire who spent three months learning the unwritten rules was a cost.

We called it something else. Organizational friction. Communication. Growing pains. "Alignment work." We invented meetings to fix it. We hired program managers to manage it. We built layers of middle management whose primary job was to translate intent from one layer of the organization to the next.

The cost was enormous and invisible because there was no audit trail.

Now there is.

When an AI system acts on ambiguous input, the failure is visible. The output is wrong in a specific, traceable way. The system did exactly what it was told and produced something nobody wanted, and you can look at the input and see exactly where the ambiguity lived. The system is a mirror that shows you what your instructions actually said, as opposed to what you meant.

A head of content at a publishing company told me in early 2026 her team had started using AI to generate first drafts of marketing copy. The early outputs were, in her words, "technically correct and totally wrong." The tone was off. The emphasis was misplaced. The copy hit every point in the brief and missed the point of the brief.

The brief itself was ambiguous. It had always been ambiguous. Her writers had compensated for years by knowing the brand, the audience, the unwritten rules. The AI had no access to any of that. It read the brief literally. The brief, read literally, produced copy nobody could ship.

The ambiguity had been costing her team for years in rework, in misalignment, in three rounds of revision every piece went through before it matched what she had in her head. Human writers absorbed the cost silently. The AI made it visible.

The pipeline was the answer in engineering because the pipeline forced precision. You could not tell the CI/CD system to "make sure the code is good." You had to define good. You had to make choices about what mattered and in what order and what you were willing to accept.

The machines could not run on vibes. They forced engineering to declare its intent in a form a system could act on, and that declaration is what made the next decade of engineering productivity possible.

The rest of the business now needs the same discipline. Not because AI demands it, though AI does. Because the cost of not doing it was always there, hiding in plain sight, and the systems you are building next will not let you hide it anymore.

Engineering figured this out more than a decade ago. The rest of the business has the option of learning it from engineering or learning it the hard way.

The next chapter names the discipline. It has a name, a shape, and a set of things you can do on Monday morning. It is what comes after the prompt.

3

CHAPTER 3

Intent Architecture

The discipline of saying what you actually want

In February 2026, the head of brand at a direct-to-consumer skincare company forwarded me a fourteen-page voice and tone document and asked why the AI-generated copy was still wrong. The document had a CMO sign-off page. It had sections on brand personality, audience archetypes, emotional register, and "guardrails for creative expression." It was beautifully formatted.

It did not define the voice. It did not define the tone.

It said "We speak with warmth and authority." It said "Our tone is confident but never arrogant." It said "We meet our customer where she is." None of those sentences are wrong. All of them are useless to a system that has to produce an actual sentence on a product page at three in the morning. Warmth is not a specification. "Confident but never arrogant" is a bumper sticker. "Meet our customer where she is" requires knowing where she is, which the document never addressed.

The team had given the system their voice and tone document. The system was following it. The output was wrong.

The document existed. The intent did not, at least not in any form a machine could act on. The gap between the existence of the document and the absence of the actual intent is the gap this book is about.

What intent architecture is

Intent architecture is the discipline of writing down what your organization actually wants, with enough precision, enough completeness, and enough honesty about the trade-offs involved, that a system can act on it without a human translating every time.

Almost every word in that sentence is doing work.

"What your organization actually wants." Not what it says it wants. Not what it said it wanted in 2023. What it actually wants, this quarter, in operational terms that produce different outputs when changed.

"Enough precision" means a system reading the statement would produce materially different work than a system reading a slightly different version of it. "Grow revenue" fails this test. "Generate forty qualified demo requests per month from companies with two hundred to two thousand employees in healthcare and financial services, where qualified means budget authority and a contract renewal within six months" passes it.

"Enough completeness" matters because precision without it gives you a system that does one thing well and ignores the eight things you also needed it to account for.

"Enough honesty about the trade-offs" is the part organizations skip. Every intent involves a trade-off. Speed against thoroughness. Volume against quality. Short-term revenue against long-term positioning. If the trade-offs are not named, the system makes the trade-off for you, in whatever direction the ambiguity allows.

Engineering organizations learned to write pipelines because machines could not run on vibes. Intent architecture is the same idea, scaled up. Your systems need to act on your intent. Your intent has never been written down in a form they can read. Closing that gap is a discipline, not a doc.

How intent architecture differs from the things it gets confused with

People hear "intent architecture" and reach for adjacent artifacts they already produce. The reach is reasonable. The artifacts are not the same thing.

Artifact	What it does	Why it is insufficient alone
Mission	Explains why the organization exists	Too broad to execute

Artifact	What it does	Why it is insufficient alone
Strategy	Sets direction	Often too interpretive
OKR	Defines measurable goals	Does not resolve trade-offs
Brief	Frames a project	Usually assumes human judgment
Prompt	Instructs a model in the moment	Repeats translation every time
Intent document	Makes operational intent explicit	Must be maintained or it drifts

Each row has a legitimate function. None of them, on its own, gives a system enough to act on. A mission tells a person why they show up. A strategy tells a team where to point. An OKR tells everyone what to measure. A brief tells a project what it is for. A prompt tells a model what to do right now. An intent document connects all of them to the actual machine running underneath. The last row is the one most organizations are missing.

The functional intent document

The functional intent document is the load-bearing artifact. It has five parts. The first three were obvious from the start. The last two only appeared after leaders described to me what happens when the first three fail in production.

The what. The outcome you want, stated precisely. Not the activity. The outcome. "Generate forty qualified demo requests per month from companies with two hundred to two thousand employees in healthcare and financial services." Test for precision: could two reasonable people read this and disagree about whether a specific outcome satisfies it? If yes, it is not precise enough.

The why-now. The condition that makes this the right priority at this time. "We need to generate mid-market pipeline because our enterprise sales cycle lengthened from nine months to fourteen months in Q3, and we need near-term revenue while the enterprise deals close." My first batch of intent documents omitted this section, on the grounds that context was not content. Teams then treated the intent as permanent, and permanent intent is almost always wrong intent.

The trade-offs you have already decided. "We will prioritize close probability over deal size." "We will accept lower open rates in exchange for tighter targeting." "We will not produce content for the enterprise persona this quarter, and if enterprise deals close based on content the mid-market team produced, that is acceptable." These sentences are hard to write. They feel like closing doors. That is the

point. A system that finds an open door walks through it.

The decisions you are leaving open. A good document does not decide everything. It names what is unresolved and why. "Webinar attendees versus content downloaders, scoring weight, open through Q2." I missed this for months. I kept producing documents that felt airtight, then watching teams quietly invent their own answers for the parts the document did not cover. Saying "this is open" is itself a decision.

The conditions that would cause you to revisit. "If enterprise sales cycles return to nine months or less, mid-market pipeline targets reopen." This part turns the document from a decree into a living decision. It tells the reader, human or system, what would make the current intent wrong.

Five parts. The what, the why-now, and the trade-offs are the load-bearing structure. The open decisions and the revisit conditions keep the document honest over time.

Here is a weak version and a strong version of the same intent, side by side.

Weak. "Drive customer retention through proactive engagement and a focus on high-value accounts. We will use AI to surface at-risk customers and personalize outreach. Trade-offs will be made as needed. The team will review this regularly."

Strong. "Keep gross retention at or above ninety-two percent through Q4 for accounts above twenty-five thousand ARR. We deprioritize accounts below ten thousand ARR for proactive outreach this quarter (CS bandwidth is below plan and the bottom segment churns on price, not service). Open: whether mid-tier accounts (ten to twenty-five thousand) get the high-touch playbook or the self-serve nudges. We are testing both through October. Revisit if gross retention drops below ninety percent in any single month, or if we hire two additional CSMs."

The weak version is a paragraph of sentences each of which sounds operational and none of which constrains a single decision. The strong version names the number, the segment, the trade-off, the open question, and the trigger. A human reading the strong version knows what to do. A system reading it does too.

A teardown teaches more than another good example. A marketing operations team sent me this in late 2025.

"Our intent is to accelerate pipeline velocity and increase the quality of our top-of-funnel engagement by aligning our content strategy with the highest-value buyer journeys, while maintaining brand

consistency and operational efficiency."

Three problems.

First, the verb stack ("accelerate," "increase," "aligning," "maintaining") does all the work and none of it. None of those verbs is measurable as stated. "Accelerate pipeline velocity" has no number, no baseline, no time window. It reads as ambitious and contains no commitment.

Second, the constraints are presented as compatibilities ("while maintaining"), when in practice they are trade-offs. Operational efficiency and high-quality engagement compete. Brand consistency and pipeline velocity compete. The sentence buries the actual decisions under "while," which is the literary equivalent of pretending the trade-off does not exist.

Third, no expiration. Nothing in the sentence would tell you, six months from now, whether the intent has been satisfied, whether it should still be the intent, or what would make it wrong. It survives any quarter unmodified, because it constrains nothing.

The marketing ops team rewrote theirs over two afternoons. The new version was one page shorter and contained three numbers the old version did not.

Why leaders resist writing it down

Vagueness has a purpose, and pretending otherwise would miss the actual reason these documents are hard to produce.

A VP who keeps strategy slightly ambiguous keeps the ability to adjust without looking inconsistent. A director who does not write down trade-offs avoids the political cost of telling one team they are lower priority. A CEO who speaks in direction rather than specifics keeps room to respond to market shifts without issuing a retraction. None of that is irrational. For most of the history of organizational management, strategic flexibility was a survival skill.

What changed is that the translation layer is going away. Not completely, and not tomorrow. Directionally, and soon. When a system is the actor, ambiguity is not flexibility. It is a failure mode. The system will not absorb the inconsistency. It will execute on it, and the output will show you exactly what your ambiguity meant.

A CRO I talked with in March 2026 described a ninety-minute argument with her CFO over the same point at a Series C fintech. They had agreed, in principle, on the quarter's revenue intent. They had not agreed on which trade-off it implied.

She wanted the document to commit to mid-market growth and accept a hit on enterprise expansion ARR. Enterprise renewals were three quarters out; the pipeline coverage problem was immediate. He wanted the document to commit to enterprise expansion and accept a hit on mid-market new-logo count. Enterprise expansion was higher margin; the mid-market motion was unproven against current pricing.

She told me she said, twice, "We are not running two strategies, we have to pick." He said, twice, that picking mid-market would make the enterprise number unreachable for the year. Each was right about what the other was wrong about.

She said the room asked, near the end, whether the document could just say "balance both." A system reading that would treat them as equal and queue them in whatever order the input order suggested, which was not what either of them wanted.

The argument did not resolve. It got decided. The CEO walked in twenty minutes late, listened for three of them, and said the document committed to mid-market, with an enterprise expansion floor of fifteen percent and a revisit trigger if mid-market booked under eighty percent of plan in any month. She got most of what she wanted. He got a floor and a trigger. Neither of them looked happy. Both of them signed.

A week later, she told me, the CFO mentioned in passing that he still thought the call was wrong. He also said it was the first time in two years the company had a revenue intent he could plan against. Both things were true.

The most common objection I hear about intent documents is that they feel like predictions. Conditions change weekly, the argument goes. Why write down what you want when it will be wrong by Friday.

The objection misreads what the document is doing. An intent document is a statement of what you want to be true right now, given what you know right now, with explicit conditions under which it should change. A supply chain version might read: "Hold a fourteen-day inventory buffer for SKUs in the top twenty percent by revenue, seven days for the next thirty percent, accept stockout risk for the bottom fifty percent. Revisit if primary vendor lead times exceed twenty-one days or if Q2 demand projections shift by more than fifteen percent." That is a described state of the world with change conditions attached, not a prediction of what the world will look like. A system can act on it. The Monday meeting where a leader verbally adjusts priorities based on last week's data cannot, except by that leader's continued presence. Two hours every Monday. Fifty weeks a year. A hundred hours of translation, against intent that could have been written down once.

What comes after the document

The intent document is the starting artifact. It is not the whole operating model. Once you have the document, you need a way to connect it to work, a way to maintain it as the world changes, and a way to learn from the moments when the system gets it wrong. Trade-off registers, open decision logs, ownership rules, the maintenance cadence, execution hooks, override logs. Each of these is a real component, and each gets developed where it earns its place in the rest of the book.

For now, the discipline starts with one page.

Tool: Intent Document Template

A clean intent document is one page, occasionally two. Use these field labels exactly. Resist the urge to add sections; the discipline is in keeping it short.

- Desired outcome
- Why now
- Target audience or system or process
- Decided trade-offs
- Open questions
- Revisit triggers
- Owner
- Review cadence

Here is the template, filled, for a fictional revenue team led by a VP of Sales named Maya Chen at a Series C horizontal SaaS company.

- Desired outcome. Generate sixty qualified opportunities per month from mid-market accounts (two hundred to two thousand employees) in financial services and healthcare, where qualified means a named internal advocate with budget authority and a documented contract renewal or expansion window within six months.
- Why now. Enterprise sales cycle has lengthened from nine months to fourteen months across our last three closed-won enterprise deals. We need near-term ARR while the enterprise pipeline matures. This intent is valid for Q3 and Q4 of the current fiscal year.

- Target audience or system or process. Outbound SDR motion (Outreach sequencing tool), AI-generated personalization layer, MQL-to-SQL routing logic in HubSpot, weekly pipeline review.
- Decided trade-offs. Close probability over deal size. We will pass on accounts above five thousand employees this quarter even when they match the ICP. We will accept reply rates twenty percent below our 2025 baseline in exchange for higher SQL conversion. Marketing will not produce enterprise-tier case studies this quarter.
- Open questions. Whether we segment financial services into commercial banking versus fintech (resolution by end of Q3, owned by Maya with Marketing). Whether we add insurance as a third vertical (resolution dependent on the financial services pilot data).
- Revisit triggers. Enterprise sales cycle returns to twelve months or less (reopen the full document). Demo-to-close conversion in mid-market drops below ten percent for two consecutive months (reopen ICP criteria). The company acquires another business with overlapping customer base (reopen all trade-offs).
- Owner. Maya Chen, VP of Sales. Document lives in Notion under Revenue/Intent. Maya has sole authority to update Decided trade-offs without escalation; changes to Desired outcome require CRO sign-off.
- Review cadence. First Monday of every month, thirty-minute review with Maya, the Director of Sales Ops, and the Director of Marketing. Override log reviewed at the same meeting.

The filled version is one page. The blank version takes ten minutes to set up. The argument that produces the filled version, the part where Maya and her CRO actually decide whether close probability beats deal size, is the work. The document is what survives the argument.

4

CHAPTER 4

The Prompt Tax

Why every prompt cycle costs more than it looks like it does

A product manager I will call Derek runs a B2B SaaS company's customer feedback workflow that I have not stopped thinking about. He told me in late 2025 he had built a seven-step prompt chain that turned raw customer feedback (support tickets, NPS comments, recorded sales calls, churn exit surveys) into a prioritized list of feature recommendations for his engineering leads. He had cut a process that used to take two days down to about twenty minutes a week.

I called him again in March 2026. He had it down to fifteen minutes. Still running it. Manually. Every Monday morning, between his standup and his 10:30 product council.

I asked what would happen if he went on vacation. "I usually batch two weeks before I take time off, and I do four hours the Saturday I get back."

I asked what would happen if he left the company. He paused longer. "The chain would rot within a month. Some of the prompts have specific assumptions baked in. The second step expects the support tickets to be tagged a certain way, and we changed that tagging twice last year and I had to patch the prompt both times. I trained two of the other PMs on it. Neither of them runs it. One sent me a Slack last month asking if we still ran it."

Derek's chain works. The output is good. He is good at this. The whole thing is also a tax, and the tax has Derek's name on it. If Derek is on a beach in Portugal, the engineering leads do not get their prioritized feedback that week. If Derek leaves, the chain rots within a month and the product council goes back to running on whatever the loudest customer said in last week's QBR. The value is real. The mechanism is one person wide.

What the prompt tax actually is

The prompt tax is the recurring cost of operating a prompt-based AI workflow. It is not the cost of building the workflow. It is not the cost of the model. It is the cost of running it, week after week, paid in time and attention and organizational fragility.

The tax has six line items, and most teams I have worked with see only the first two.

The cost of translating messy work into model instructions. Every prompt is an act of translation. The person at the keyboard takes a real-world problem (a customer is unhappy, a campaign needs copy, a contract needs review) and turns it into structured language a model can act on. That translation has a price. It takes time. It takes a particular kind of mental work, the work of holding the messy thing in your head while you describe it in a form that strips out most of its mess. The person doing it gets better at it over time. The work does not go away.

The cost of reviewing and correcting output. The model produces something. The human reads it. The human catches the parts that are wrong, the parts that miss the context the prompt did not capture, the parts that are technically correct and somehow off. The human edits. Sometimes the human re-prompts. The review is not optional. The output is not trustworthy enough to skip it, and everyone running these workflows knows this even when they will not say it in a status update.

The cost of moving output between systems. The model lives in one tool. The output needs to land in another. A Slack message. A CRM field. A Jira ticket. A document somewhere. Copy, paste, format, attach. This is the part everyone underestimates because it feels like nothing. Fifteen seconds here. Twenty seconds there. Across a week of prompt workflows, it adds up to real time, and it adds up to errors, because every paste is a chance to put the wrong thing in the wrong place.

The cost of remembering to run the workflow. Derek's chain runs Monday morning because Derek runs it Monday morning. If Derek forgets, the chain does not run. The team does not get the prioritized feature list. The engineering leads do not have the input they have built their week around. The cost of remembering does not show up on any timesheet. It shows up in the cognitive load Derek carries from Sunday night through Monday morning, every week of the year.

The cost of expert dependency. Derek is good at this. The chain works because Derek built it and Derek runs it and Derek knows which steps need attention this week and which can be run on autopilot. If a colleague tries to run it cold, they get a worse result, because the chain encodes assumptions Derek made and never wrote down. The expert dependency is the tax on the team for having a workflow that lives inside one person's head.

The cost of workflow decay. This is the line item that surfaces last and hurts most. When Derek leaves, the chain rots. Some of the prompts stop working because the data sources change. Some of them stop being relevant because the product changes. The PMs who inherited the chain do not understand the assumptions well enough to update them, so they patch around the broken parts, and within six months the chain produces output nobody trusts. By the time anyone calls it dead, it has been functionally dead for a quarter.

Six costs. Most teams account for the first two. The other four are paid by the same person who paid the first two, which is why nobody notices them until that person goes on parental leave or takes another job.

The number behind the interruption

Gloria Mark's lab at UC Irvine has been measuring what interruptions actually cost since the early 2000s. The number that survives every retelling is twenty-three minutes and fifteen seconds. That is the average time it takes a knowledge worker to return to full focus on a task after being pulled away from it.

A prompt cycle is an interruption. It is a useful one. The worker stops what they are doing, opens a tool, translates the problem, reviews the output, moves the output somewhere, and returns to whatever they were doing before. The recovery time on that return is not zero. It is, on average, closer to twenty-three minutes than to the two or three minutes the worker thinks it took.

Derek runs his chain Monday morning. He plans for fifteen minutes. The actual cost to his attention, measured the way Mark's lab measures it, is closer to forty. He does not feel it as forty, because he is good at this work and the chain is interesting to him. The calendar does not care how it feels.

Multiply across a team. Twelve people on a marketing team, each running three or four prompt workflows a week. Forty to fifty interruptions across the team in a typical week, each carrying a recovery tail. The math is not pleasant.

Prompt-based versus intent-based, made explicit

I want to set the contrast plainly, because everything in Part III turns on it.

A prompt-based workflow is one where a human repeatedly instructs the machine. Each cycle requires the human to be present, to translate, to review, to move output, to remember. Derek's chain is prompt-based. The skincare team's content workflow is prompt-based. Most of the AI deployments I have seen in 2026 are prompt-based, regardless of how sophisticated the prompts have become.

An intent-based workflow is one where a system acts against maintained organizational intent and asks for human judgment only at exception points. The intent document from Chapter 3 is the input the system reads. The system does the translation work continuously, against the document, and produces output that aligns with the stated intent without a human re-translating each cycle. The human gets pulled in for the exceptions: cases the document does not cover, edges the system is unsure about, decisions that would change a trade-off and therefore need to escalate to whoever owns the document.

The difference is not about how good the prompts are. The difference is about where the translation happens. In a prompt-based workflow, the translation happens at the moment of each use, and a human does it. In an intent-based workflow, the translation happens once, at the moment the intent is written, and the system does it from then on. The human only re-engages when the intent itself needs to change.

The same distinction shows up in the time profile of the work. A prompt-based workflow has flat recurring cost: roughly the same minutes every week, every quarter, every year. An intent-based workflow has a high upfront cost (writing the document, hooking it into systems, building the override log) and a low recurring cost (running the cadence, reviewing overrides, updating when triggers fire). The total cost of the intent-based workflow over a two-year horizon is almost always lower, often by a wide margin. The total cost over a six-week horizon is higher, which is why most teams never make the switch. They are optimizing the wrong window.

This sounds like a small architectural distinction. It is the entire game.

What I got wrong about prompt templates

For about a year and a half of those two years, from mid-2024 through late 2025, I was telling people in every conversation that the answer to the prompt tax was better prompt templates. Reusable, parameterized, version-controlled prompts a team could share and improve. I wrote about a few good ones. The templates worked. They cut translation time. They standardized output quality. They made the tax slightly lower.

I was wrong.

The template was a faster version of the same translation tax, not a different model. Every cycle still required a human to fetch the template, fill in the variables, send the prompt, review the output, and move the result. The template made each step take less time. It did not eliminate any step. The recurring nature of the cost was unchanged.

I caught it in October 2025, on a call with a customer success director who walked me through a template library her team had built for renewal outreach. Forty-seven templates. Sharp, parameterized, version-controlled in a shared Notion database with usage analytics. She showed it off with real pride. I asked how much time her team was spending operating the templates each week. She did the math out loud. About six hours per CSM, across a team of eight. Forty-eight person-hours a week. About twenty-five hundred hours a year.

The templates had reduced the time per cycle. They had not reduced the number of cycles, and they had not changed the fact that every cycle required a human in the loop. The tax rate was lower per transaction. The volume of transactions had grown. The total tax was the same or larger than before they built the library. The team had built a more efficient version of a workflow that was the wrong shape.

What I had been advocating, when I look back at the slides I used in 2024, was a faster horse. The templates were the equestrian equivalent of better saddles, lighter shoes, a more efficient feeding schedule. They produced a faster horse. They did not produce a car.

I changed my framing after that conversation. The right move was an intent document the system could act on, with the templates relegated to a fallback role for the cases the document did not cover. She told me later that her team rebuilt their workflow around an intent document over six weeks. They kept maybe a dozen of the original templates. The rest got retired. The team's weekly hours on renewal outreach dropped from forty-eight to about fifteen, and the CSMs reported that the work that remained was the work that actually used their judgment, not the work of operating the templates.

I do not regret the template work. It was the right move with the tools we had at the time, and it taught the teams that built templates a vocabulary they used later when they wrote intent documents. Knowing how to parameterize a prompt is part of knowing how to write an intent document, because the parameters are the variables and the document is the schema. The template phase was not wasted. It was just not the destination, and I spent eighteen months telling people it was.

Toward drift

Even when you write the intent down once, the document goes stale. The customer changes. The competitor changes. The team changes. The product changes. The document does not change unless someone changes it, and the person who would change it is busy doing the work the document was supposed to make possible.

The intent document, written once and never revisited, is the next failure mode.

Tool: Prompt Tax Calculator

Calculate the prompt tax for any recurring workflow by listing the six cost categories and putting a real number on each. The first four are per-cycle costs. The last two are organizational risk costs that show up as discrete events.

Here is a worked example, using Derek's customer feedback chain.

- Translation. Each cycle: roughly four minutes to gather the inputs, structure the prompt context, and configure the chain. Fifty weeks. Two hundred minutes a year. Plus an attention recovery tail of about ten minutes per interruption, applied weekly. Five hundred minutes a year on translation alone.
- Review. Each cycle: roughly six minutes to read the output of each step, catch the wrong calls, re-run two of the seven steps when the output is off. Three hundred minutes a year.
- System-to-system handoff. Each cycle: roughly two minutes to move the output into the team's Linear board, tag the engineering leads, and update the PM tracking sheet. One hundred minutes a year.
- Remembering to run. Hard to quantify, but real. Derek estimated he spent five minutes each Sunday evening mentally rehearsing the Monday morning run, plus another five minutes Monday morning re-orienting before starting. Roughly five hundred minutes a year.
- Expert dependency. Discrete cost. Derek's two-week vacation produced a four-hour Saturday catch-up session and a noticeable gap in the engineering team's feature prioritization the second week. Annualized, call it one full working day, eight hours, four hundred and eighty minutes.
- Decay. Discrete cost, paid when Derek leaves. Conservative estimate, based on what I have heard about other companies: six weeks of degraded output across the team while a replacement learns the chain, plus the cost of either rebuilding the chain or accepting permanently worse customer feedback synthesis. Sixty hours of replacement effort plus an unmeasured loss of quality. Thirty-six hundred minutes for the rebuild, with the quality loss carried separately.

Add the recurring costs: five hundred plus three hundred plus one hundred plus five hundred is fourteen hundred minutes a year. About twenty-three hours of Derek's time, every year, just to operate the chain. Add the dependency cost of four hundred and eighty minutes. The recurring annual tax is about thirty-one hours per year of Derek's working time, before accounting for the decay event that will eventually arrive.

Thirty-one hours of a senior PM's time, per workflow, per year, for one prompt chain. Derek runs three other recurring AI workflows. Multiply.

The calculator does not tell you whether the workflow is worth running. Many of these workflows are clearly worth their tax. The calculator tells you what you are paying, so when you compare the prompt-based version to an intent-based equivalent, you have a real number on both sides.

5

CHAPTER 5

The Drift Problem

Why the intent document you wrote last quarter is already wrong

A VP of Customer Success I spoke with described a strong intent document her team had built in June 2025. Eight CSMs at a B2B software company. The document specified renewal priorities, escalation criteria, trade-offs between gross retention and expansion, ARR tiering. Revisit conditions. An owner. A review cadence. She signed off. The team began building automated workflows against it.

In December, the company acquired a smaller competitor. The customer base nearly doubled. The team grew from eight CSMs to thirteen. The acquired customers used a different pricing model. The roadmap shifted to integrate the two platforms. She was reassigned to lead the CS integration, and the function spent the first quarter of 2026 in organized chaos.

The document was still pinned in Confluence. The workflows were still running against it. Nobody had touched either one.

By April, when she walked me through what had happened, the document was fiction. The acquired customer base was not in the ARR tiering. The trade-off section still ranked gross retention above expansion ARR, but the acquisition had made expansion the most urgent metric of the integration period. The qualification criteria in the document were being applied to customers who had signed on a pricing model the document had never considered. The product changes had invalidated three of the

escalation triggers. The workflows were producing outputs that made sense for the pre-acquisition company and were actively wrong for the post-acquisition reality. CSMs had been working around the system manually for four months.

I had this wrong for a long time. I had been telling leaders to treat the intent document as an artifact to deliver, not a living record to maintain. That was my mistake, and I have heard about it at enough organizations to call it the default failure mode rather than the exception.

What drift is and why it is the worst of the three

The first two failure modes are vagueness and incompleteness. Vagueness means the document never said the thing precisely. Incompleteness means it said the precise thing but never named the trade-offs. Both are catchable in the writing phase, if the writer knows to look.

Drift is different. Drift means the document was correct when written and is no longer correct now. The world changed. The document did not. The system reads it and acts confidently on instructions that no longer describe what the organization wants.

Drift is the quietest of the three and often the most expensive. Vagueness produces output that is obviously off, and the team notices fast. Incompleteness produces output that is wrong at the edges, and a sharp reviewer catches it. Drift produces output that looks right by every test the system runs, because the system is testing against the original document, and the original document is what is wrong. The longer drift goes uncaught, the more decisions are made on stale intent, and the more those decisions compound into a strategy nobody chose.

The three shapes of drift

Drift shows up in three forms. I have not seen a fourth.

Priority drift is when the relative weight of priorities has changed but the document still ranks them the old way. The CS team's document still put gross retention above expansion ARR after the acquisition had made expansion the urgent metric. Strategy decks and all-hands talks reflect that kind of shift within weeks, while the intent document tends to lag behind. The system, reading the original, keeps treating the old top priority as the top priority, and people work around the gap manually.

Definition drift is when the words still appear correct but the meaning has changed underneath. "Qualified account" meant one thing in the CS team's original document and a different thing once the acquired customers were inside the ARR tiering on a different pricing model. The document never

noticed the word now carried two definitions. Two people in the same meeting can read the same sentence and disagree about what it requires, because they hold different definitions of the same word.

Scope drift is when the territory the document covers has changed. The CS team's document, written for eight CSMs and four hundred accounts, was still claiming to govern the function after the team had nearly doubled. The half of the function that did not exist when the document was written was operating outside it entirely.

The three forms compound. A team can have priority drift, definition drift, and scope drift at the same time, and each layer makes the others harder to see. The CS team had all three. The acquisition triggered each one, and the document had no language to flag any of them.

What sits next to the document

Catching drift requires more than reading the intent document. It requires three artifacts alongside it.

The trade-off register. The document states decided trade-offs in the present tense: this beats that, this quarter. The register is the history. When the document says "close probability over deal size," the register says when that call was made, who made it, and what conditions produced it. Drift in a trade-off is often invisible because the document just shows the current answer. The register shows the answer was last revisited fourteen months ago, under conditions that no longer apply. A revenue leader I spoke with described a clean document and no register. When her team tried to reopen the close-probability trade-off in Q1 2026, no one could remember why the original call ranked it above deal size, which meant the conversation reopened everything, including settled questions nobody had wanted to relitigate.

The open decision log. The explicit list of what the document has not decided, and why not. Without it, teams quietly invent their own answers for the gaps. A system operating against a document that does not say what is open will treat everything as decided. A human will fill the gaps with assumptions and remember the assumptions as policy.

The ownership rule. Not who wrote the document. Who is allowed to change which parts of it, and what gets escalated. The CS team document had the VP as owner. It did not say which sections she could change unilaterally and which required CRO sign-off. When she was reassigned, nobody knew whether they had the authority to edit, and the safer move was the wrong one. Without an ownership rule, edits stop happening or happen by whoever feels strongest, and "feels strongest" is not a governance model.

The maintenance model

The maintenance model has two components. Both are needed. Neither is sufficient alone.

The first is cadence of review. A named person, with explicit ownership, reviews the document on a fixed schedule. Monthly for operational functions like outbound sales and customer support. Quarterly for strategic functions like pricing and product roadmap. The review is not a rewrite. It is a check. Three questions: Does this still describe what we actually want? Has any revisit condition fired? Has anything changed that the conditions did not anticipate? Twenty minutes if the answers are easy. A half-day if the third question opens something up.

The second is trigger conditions for revision. These are events named in the document itself, the "if enterprise sales cycles shorten" or "if Q2 demand shifts by more than fifteen percent" lines. When a trigger fires, the document gets reopened. Not just reviewed. Reopened, which means the trade-offs get re-examined and the decisions get remade if the underlying conditions warrant.

The cadence catches the slow drift. The triggers catch the sudden changes. Most documents that fail in production fail because they had one and not the other. A team with only triggers misses the slow accumulation no single trigger captures. A team with only cadence catches the slow drift but misses the discrete event that should have reopened the document the day it happened.

The CS team had a quarterly cadence and three triggers. The triggers were good. None of them named "the company acquires a competitor." It is the kind of event that feels obvious in retrospect and is invisible when you are writing the document for a company that has never made an acquisition.

I now suggest every intent document carry a catchall trigger: "If anything happens that materially changes the customer base, the product, the team size, or the strategic direction, this document is reopened regardless of cadence." That phrasing is loose by design. The point is to give the owner explicit permission to call a reopen even when no specific trigger has fired.

When the cadence stops

A maintenance model fails most often the same way. Not because the cadence was wrong, but because something disrupted it and nobody noticed it had stopped. A leadership change. An owner reassigned. A meeting moved once and never put back on the calendar. The document keeps being read by the system. The review never happens. Nine months pass before someone notices the workflows have been acting on stale intent the whole time, and by then the recovery is not the rewrite (which takes an afternoon) but the months of decisions that have to be reopened, the overrides that need explaining, and the team's trust in the system that needs resetting.

The maintenance model needs a meta-check: who notices when the maintenance stops happening. I now recommend every intent document name a second person, usually the function lead one level up, who receives confirmation each cadence that the review happened. If the cadence is missed twice, that person is responsible for restarting it or naming a new owner. Without the meta-check, the maintenance loop dies quietly. The cost only surfaces after the damage.

When the document is good and the result is still bad

A bad outcome from a good document teaches more than a bad outcome from a sloppy one.

Imagine a strong intent document at a healthtech company. Desired outcome: a hundred and ten qualified opportunities per month, defined. Trade-offs named: paid social over events, mid-market over SMB. Triggers and ownership in place. The document is hooked into the agency brief, the SDR sequencer, and the content pipeline.

Now imagine paid social CPMs in the two core verticals doubling between February and April. The team holds the volume target through March and April by spending against the budget cushion the original plan built in. The cushion runs out in May. The number comes in at sixty-eight. Below plan by almost forty percent.

The document is not wrong. The shift was outside the trigger list because CPM volatility was not the kind of thing anyone put on a trigger list in February. The system keeps executing against the document. It allocates budget against the named trade-offs. It produces the pipeline the document tells it to, at the cost the market charges, and the math no longer works. The team sees the underperformance in week three. They do not reopen the document until week eight, because no trigger has fired. The intent document, perfectly maintained, becomes the reason the team did not change course earlier. The discipline makes them confident in the wrong thing for five extra weeks.

No list of triggers is exhaustive. The owner has to be willing to call a reopen on instinct when the numbers stop matching what the document predicts. The intent document does not relieve the leader of judgment. It tells the leader where to apply it.

Failure modes inside the framework

Even with the document, the register, the log, the ownership rule, and the maintenance model, the system can still fail in specific ways. I have seen six.

Over-specification. Intent written so tightly that teams lose room for judgment. The document constrains good calls, not just bad ones. The skincare team in Chapter 3 swung the other way after their fourteen-page voice doc failed and wrote a thirty-page replacement naming verb counts and syllable maxes. The copy hit every rule and read like instruction manuals.

False precision. Numbers that look rigorous and are not tied to decisions. "Increase mid-market activation by 27.3 percent." The decimal point reads as data and is decoration, because nothing in the system changes based on whether the number is 27.3 or 22 or 31. I heard about a sales ops team that spent six weeks instrumenting a dashboard for a 27.3 percent target that had been pulled from a back-of-envelope projection.

Ownership theater. Someone named owner with no authority to change the document. The CS team after the acquisition had a named owner whose name was still on the document and who had been reassigned to a different function. The role existed in the artifact, not in the org chart. Without authority that matches the name, the ownership rule is a label, not a control.

Review theater. The monthly review happens. The thirty minutes get spent. Nobody reopens the real trade-offs, because reopening them is uncomfortable and the room has agreed, without saying so, to avoid the discomfort. The cadence becomes a ritual that protects the document instead of stress-testing it. Leaders have described these to me where I could tell by minute fifteen.

Execution disconnect. The document exists. The systems do not read it. Trade-offs are written down in Notion and the outbound tool is running the previous quarter's rules. The document still says the right thing and the system still runs without error. The only way to catch this is to trace, every quarter, the path from each line of the document to the system that should be acting on it.

Override blindness. Humans keep overriding the system. The overrides get logged or do not. Either way, nobody studies the pattern. The healthtech example illustrates the cost: the CPM problem surfaced in week three at the SDR and agency level, where reps were already rerouting budget away from the channels the document named as priorities. The team did not reopen the document until week eight. The signal was there. Nobody was reading the overrides as a class.

Six failure modes. Intent architecture does not make problems disappear. It makes them visible. The visibility is the value, and only if someone is looking.

Why drift makes the rest of the book matter

Every chapter in Part III is a function-specific application of intent architecture. Revenue. Creative. Customer success. Operations. Each assumes the function has done the work of writing intent down, building the register, naming the triggers, hooking the document into the systems that need it.

Without the maintenance model, every application turns into the same problem: a document that was correct when written and quietly stops being correct. The team works around it. By the time anyone notices, the cost is paid in decisions that compounded for months.

The applications in Part III are not difficult to start. They are difficult to keep. The discipline is in the keeping.

Tool: Drift Review Checklist

Run this every cadence, and every time a trigger fires. The questions are blunt by design.

- Does this document still describe reality? Read it as if you have never seen it before. If you encountered it on a wiki page today, would you recognize what your team does?
- Has the strategy changed? Compare the stated outcome and why-now against the most recent all-hands deck, board update, or executive offsite output. If the strategy has shifted and the document has not, that is priority drift.
- Has the customer changed? Pull the last twenty closed-won deals (or active customers, or active users, depending on the function). Do they match the target? If the document says mid-market financial services and the last twenty are split across healthcare and logistics, that is scope drift.
- Has the team changed? Headcount, structure, named roles. A document that referenced "the CS team" when the team was six people is not the same document when the team is fourteen.
- Has the system changed? The execution hooks. If the document is supposed to feed an outbound tool that has been replaced, or a CRM workflow that has been refactored, the hooks are broken and the document is no longer running anywhere.
- Are people working around the document? The loudest signal of all. Ask the people doing the work whether they consult it, ignore it, or override the system decisions it drives. If overrides are routine, the document is already fiction. The team has moved on. The artifact has not.

Six questions. If three or more answers indicate drift, reopen the document. Do not just edit it. Reopen the underlying decisions. Drift this deep usually means the trade-offs themselves need to be remade, and an edit that does not touch the trade-offs is just rearranging the fiction.

6

CHAPTER 6

Revenue Without the Repeated Ask

How commercial teams hit the ceiling first

Forty-seven thousand outbound emails in ninety days. That was the number a growth leader at a mid-stage B2B payments company described to me in March 2026, and his team had been proud of it. Before the AI rollout they had been sending twelve thousand a quarter. Now the system was writing subject lines, personalizing openers from LinkedIn and technographic signals, sequencing follow-ups, and doing all of it without the SDR team lifting much more than a finger to approve.

Reply rate before AI: 3.2 percent.

Reply rate after: 1.1 percent.

The math was technically favorable. A 1.1 percent reply rate on forty-seven thousand emails produces more raw replies than 3.2 percent on twelve thousand, and the team had pointed this out. They were correct.

They were also losing.

The first six weeks felt like a breakthrough. Activity metrics went vertical. The SDR leaderboard, which used to celebrate the top sender at 400 touches a week, now had three reps over 1,500 and one over 2,000. Then the signal decayed. Reply rates dropped. The leads that did reply were harder to close.

Conversion from meeting-booked to opportunity-created fell from 38 percent to 22. Win rates fell by nearly a third over two quarters. The CRO looked at the numbers in a QBR and said, "we're busier than we've ever been and the revenue line doesn't know it."

At first the team thought they had a sales execution problem. They didn't.

They had an intent problem. The system was producing more outreach to more leads with more personalization, which was what it had been asked to do. Nobody had asked it to produce better outreach to the right leads. "Better" and "right" had never been defined in terms the system could act on.

The obvious response, the one I pushed for a year, was to do the strategy work first and deploy the tools second. The sequence does not work. The strategy work often cannot happen first, because the team does not know what questions to ask until the tools expose the ambiguity. The payments company did not know their ICP was vague until the AI started acting on it at scale. The vagueness was invisible at human speed. At machine speed, it became the dominant signal.

The right sequence, I now believe, is: deploy, hit the wall, then do the intent work the team was always going to need. The wall is information, not failure.

What "Ideal Customer" Means When You Have to Write It Down

The payments company had an ICP document, the growth leader told me. Two pages, sitting in their sales enablement folder. It said their ideal customer was "a mid-market or enterprise financial services company with a digital transformation initiative and a need for modern payment infrastructure." Company size ranges. Revenue bands. A few technology keywords.

He told me he later asked his VP of Sales and his VP of Marketing to independently write down, on an index card, the three attributes that most reliably predicted whether a deal would close.

The VP of Sales wrote: "Champion with budget authority. Active RFP or vendor review. Previous relationship with someone on our team."

The VP of Marketing wrote: "Growth-stage fintech. Series B or later. Currently using a legacy payment processor they've outgrown."

These two lists share zero items.

I have heard a version of this story from eight commercial leadership pairs in the past eighteen months. In six of those eight, the sales leader and the marketing leader had been producing lists with one or

fewer overlapping attributes. They had been using the same term, "ideal customer," to describe different companies, for years. The ICP document was vague enough to contain both definitions, and the humans had been absorbing the disagreement through hallway conversations and the weekly pipeline review, where someone would say "that one's not really our kind of deal" and everyone would nod without defining what "our kind" meant.

The AI could not absorb it. It read the ICP document and produced outreach for every company that matched the stated criteria. The personalization tokens it pulled from LinkedIn were technically correct (right title, right company size, right industry tag) and structurally hollow, because the document it was personalizing against did not know which of those signals mattered.

The quality decay is not a model problem. The model is doing exactly what it was told. The decay is the visible shape of a definition that was never tight enough to operate on. Humans had been quietly tightening it, deal by deal, for years. The AI did not have access to that quiet tightening.

The Week

The fix took the team a week, the growth leader told me later. Not a week of someone's spare time. A dedicated week.

Monday was alignment day. Both VPs, plus the CRO and two senior AEs, sat in a room and listed every attribute they personally used to evaluate a lead. Not what the document said. What they actually looked at. The list had twenty-three items. Some were in the ICP document. Most were not. Several contradicted each other.

Tuesday was data day. They pulled eighteen months of closed-won and closed-lost deals and looked at which attributes correlated with wins. Company size, which the ICP document treated as a primary filter, had almost no predictive value once you controlled for buying-process maturity. The strongest predictor was whether the prospect had a dedicated payments team. That was not in the ICP document. Second strongest was whether they had evaluated a competitor in the last twelve months and decided not to buy. Also not in the document.

The data pull broke something else. The team had assumed their fintech segment was their highest-performing vertical because it produced the most pipeline. On closed-won rates, fintech was third, behind insurance and logistics. The VP of Marketing had built her demand gen strategy around fintech. She did not immediately accept the result. She argued, credibly, that the fintech pipeline was younger and had not had time to mature.

They spent the afternoon trying to re-run the analysis controlling for pipeline age, realized their CRM data was not clean enough to do it reliably, and ended the day with an open question. They had to make a decision Thursday without the data they wanted. I had been telling people in conversations like this to wait and clean the data. That was wrong. Waiting would have stalled the whole week, and the decision they eventually made, to keep fintech as a secondary segment while leading with insurance and logistics, turned out to be directionally right even without the cleaner analysis.

Wednesday and Thursday was the trade-off conversation. The data showed their highest-converting leads were companies with dedicated payments teams that had recently rejected a competitor. The VP of Marketing pushed back. She argued that targeting companies that had already rejected a competitor was fishing in a depleted pond. The VP of Sales disagreed. He said those companies had already educated themselves on the category and were further along in their buying process.

The CRO made the call Thursday morning. They would prioritize the rejected-competitor segment for the next two quarters, with a secondary track for companies with dedicated payments teams that had not yet entered a buying cycle. The VP of Marketing was not happy. She said so. The CRO acknowledged her concern and said they would revisit if the secondary track's conversion rate exceeded the primary within sixty days.

Friday was documentation day. They wrote the intent document. Not a new ICP PDF. An intent document with the structure from the previous chapter: the what, the why-now, the decided trade-offs, the open questions, the revisit conditions. Four pages.

The following Monday, they reconfigured AI outreach against the new criteria. Within six weeks, reply rates climbed from 1.1 percent to 2.8 percent. Not back to the pre-AI baseline of 3.2, but on dramatically larger volume. The leads that replied were closing at nearly twice the rate of the previous quarter's pipeline.

The system did not get smarter. The instruction got smarter.

I had the data part wrong in my advice for a long time. I expected a clean data pull would settle the argument. It did not. The data tells a team what has worked in the past. It does not tell them what will work going forward, and two VPs can have different theories about the future of their market. The data is necessary and not sufficient. In revenue work, the data narrows the disagreement but a person with authority has to close it, and they have to close it in a way that gets written down so the next argument can start from a different place.

What the Second Quarter Showed

By the end of Q2, the picture got more complicated. Reply rates held at 2.8 percent. Win rates on the new pipeline were up. The CRO sent the board a chart that pointed in one direction.

Then the deal desk lead, a quiet operator named Theo, surfaced something he had been tracking on his own. Average deal size on the rejected-competitor segment had dropped fifteen percent versus the previous year's mix. The segment closed faster and at higher rates, but the deals were smaller. The companies that had already evaluated a competitor and walked away were, on average, more cost-sensitive than the companies the old ICP had been catching by accident. The old vague ICP had been pulling in a small but consistent stream of larger, slower deals that the new intent document explicitly deprioritized, and some of those slower deals had been the biggest contracts of the prior year.

The CRO did not roll back the document. He added a third track for high-ACV accounts that did not match either primary segment, with explicit criteria for what qualified one in. The trade-off was not solved. It was named, and the named version produced a smaller second argument that the team could have without anyone losing face.

A Stale Document, Five Months In

The point of the intent document is not that it stays right. It is that you can tell when it stops being right. A different team, briefly, to show what happens when nobody is watching.

A revenue ops lead at a Boston-based vertical SaaS company in the dental-practice management space described a problem to me in October 2025 her team could not explain. Their AI lead-scoring model had been rating their inbound pipeline cleanly. Close rates on the top-scored leads had been deteriorating for five months without anyone noticing. The model had not changed. The scoring rubric had not changed. The leads coming in had not changed in any obvious way.

The thing that had changed was outside the model. Their largest competitor had been acquired in May 2025 by a private-equity buyer that immediately raised prices by roughly thirty percent. Dental practices currently under that competitor's contract had become, almost overnight, the highest-intent prospects in the market. None of this was in the scoring rubric, because the rubric had been written in early 2024, when that competitor was the cheapest player in the segment and being a customer of theirs was actually a negative signal. The score still treated those accounts as low-priority. The SDR team, working from the score, called them last. Two of those accounts had closed with a different competitor in August. One had a forty-thousand-dollar ACV and had been on a verbal handshake with the company's founder a year earlier.

The model was doing its job. The job had changed. The intent document for that scoring rubric, she told me, had a revisit-conditions line that read "if market competitive dynamics shift materially." The condition had been satisfied for five months. Nobody owned checking it.

She rewrote the revisit condition that week to name the specific competitors whose pricing should trigger a review, and assigned the check to her Monday standup. It took her about forty minutes. The four lost deals were not coming back.

Before / After: outbound prompt

Before (prompt-based):

"Write outreach to fintech companies that might need payment infrastructure."

After (intent-driven):

"Prioritize companies with dedicated payments teams, recent competitor evaluation, renewal pressure within six months, and budget authority identified. Deprioritize companies without implementation urgency unless strategic account value exceeds $X."

The prompt asks for a product. The intent statement gives the system criteria it can act on without supervision, including what to leave alone.

Forecasting Activity Versus Forecasting Intent

Most AI pipeline forecasting tools predict what salespeople will do. They project historical activity forward. They do not tell you what should happen.

The payments company had an AI forecasting tool that was, by any technical measure, accurate. It had predicted quarterly revenue within 6 percent of actual for three consecutive quarters. The CRO had trusted it. The forecast reflected the existing pipeline, which had been built on the old, vague ICP. It was accurately predicting the output of a system that was targeting the wrong companies. A precise measurement of the wrong thing.

After the intent work, the growth leader told me, the CRO had a question that turned out to be hard. "Should I trust the forecast less now?" The honest answer was yes, temporarily. The old forecast was calibrated to the old pipeline. The new pipeline would take two to three quarters to recalibrate. He had not liked that answer.

There is a difference between a forecast that tells a CRO "your reps will probably close $4.2 million next quarter" and a system that says "given your stated intent to close twelve deals in the rejected-competitor segment at an average $180,000, here are the seven pipeline gaps that need to be addressed by week six." The first is a prediction. The second is a plan with visible dependencies. The payments company built a manual spreadsheet that compared their pipeline weekly against the intent document's targets, segment by segment. It was ugly. It worked better than the AI forecast for the first quarter after the transition, because it measured progress toward what they wanted.

When the CRO Is the Bottleneck

The CRO at the payments company was good. Strong product knowledge, strong relationships across the sales team, a reliable instinct for which deals were real and which were mirages. When he was in the weekly pipeline review, the team made better decisions. When he was traveling, or in board prep, the decisions degraded. Not catastrophically. Noticeably.

His judgment was the operating system. The AI tools, the CRM workflows, the scoring models, all of it ran on top of a layer of unwritten rules in his head. None of which was documented. The team had learned to read him the way a good crew reads a captain. The metaphor only goes so far. A captain on a ship is still in the room. The CRO was often not.

This worked at thirty people. It was breaking at ninety. New reps could not absorb his judgment fast enough. The scoring model did not know he viewed deals with a specific competitor as worth fighting only if the prospect had an implementation deadline within four months. The outreach system did not know he deprioritized one sub-industry because he had watched three deals in that space churn within a year, though he had never written a policy about it. He just steered away in conversation, and the experienced reps had picked up the pattern.

The CRO had resisted writing it down, the growth leader told me. He had not said "I don't want to." He had said "it's more complex than a document can capture." He had been partly right. His judgment included things that are genuinely hard to specify: a feel for timing, an instinct about which prospects would be honest about their budget and which were using the sales process as a negotiating tool with their existing vendor.

Hard is not impossible. The CRO eventually sat down with the team's head of revops over two afternoons to work through his deal evaluation heuristics. They captured the majority of his decision categories in written form. A meaningful portion remained inaccessible. The written portion was enough for the AI systems to make materially better decisions without him in the room, enough for new reps to ramp faster, enough that Monday pipeline review shrank from ninety minutes to forty-five.

Then the CRO had a moment, the growth leader said, that surprised the CRO more than it surprised the rest of his team. Seeing his own judgment written down made him realize some of it was outdated. The sub-industry he had been steering away from. He checked the data. The three churned deals were all from 2023. The churn had been caused by a product gap that had since been closed. He had been carrying a bias based on a problem that no longer existed, and he would never have caught it if the judgment had stayed in his head.

Externalization does not just give the system access to the leader's judgment. It gives the leader access to their own judgment in a form they can examine. You cannot audit what you cannot see.

The payments company is still early in this work, he told me. The CRO overrides the system regularly, and some of those overrides are good calls the document does not yet capture. Each override is a signal that the document is incomplete, and the team has started treating it that way: logging the override, examining the reasoning, updating the document when the reasoning holds up.

The maintenance model matters more than the initial document. The team assigned a single owner (the CRO's chief of staff) whose job included reviewing override logs weekly and proposing amendments monthly. The CRO did not have to remember to maintain it. The maintenance was someone's job.

That is the difference between intent architecture as a one-time exercise and as an operating practice. The exercise produces a document. The practice produces an organization whose stated beliefs are kept current with its actual beliefs. Most teams do the exercise. Few build the practice.

Revenue is the place organizations usually hit the ceiling first, because outbound and pipeline are the loudest functions. The CRO cannot hide a decaying win rate behind a vague ICP for long. The numbers force the conversation.

Creative work hits the ceiling differently. The signal is quieter. A brand that drifts off voice does not show up in a board deck for two quarters. The customer noticing the drift will not write in to complain. They stop trusting the publisher, and the trust loss looks, in the metrics, like a slow decline in newsletter engagement that no one can quite explain. The resistance comes from a more defensible place, too. Sales leaders mostly want the help. Creative leaders are afraid the help will hollow out the thing they

have been protecting.

7

CHAPTER 7

Creative Work Without Creative Collapse

Constraints, not prescriptions

"Every time someone shows me a system that can write like us, I ask them to show me a system that knows when not to."

A brand director at an athletic apparel company said that to me in February 2026. Nobody had an answer for her.

Sales teams worry about speed. Creative teams worry that the act of specifying what they want will kill the thing that makes it good.

This worry is correct, in one specific domain. Over-specification of aesthetic output destroys creative quality. Tell a designer exactly what the layout should look like, pixel by pixel, and you have removed the designer from the process. Tell a writer exactly what sentences to produce and you have a transcriptionist. The trouble is that this gets treated as the entire argument against bringing intent architecture into creative work. It isn't. The full argument requires a second step. Separating where specification belongs from where it doesn't.

Creative teams have watched what happens when other parts of the organization try to codify their work. Style guides that run to two hundred pages and produce output that is grammatically clean and completely lifeless. Brand workshops where consultants reduce the voice to four adjectives on a

quadrant chart. AI-generated first drafts that match the brief perfectly and read like they were written by someone who has never purchased anything, felt anything, or been confused by anything. The resistance looks like stubbornness from the outside. It is quality protection.

The obvious response is to write better briefs. Be more precise. Watching several creative teams try that response showed me why it fails. The problem is not imprecision. The problem is specifying the wrong layer. When you specify the output layer, the aesthetic surface, you get compliance. When you specify the intent layer, what must be true about the work regardless of its form, you get creative freedom operating inside borders that actually help.

The Brief as Intent Document

A brief that says "the headline should be short, punchy, and use an active verb" is a prescription. A brief that says "the reader must know within three seconds what this product replaces in their current workflow" is a constraint.

Prescription tells the creative person what to make. Constraint tells them what the work has to accomplish. Prescription closes options. Constraint opens them.

Most organizations have lost the distinction, if they ever had it. Briefs accumulate prescription over time. Someone writes a brief that says "the tone should be warm." A year later, someone adds "warm means conversational, with short sentences." Two years after that, someone adds "short sentences means under fifteen words, with no semicolons." By year three, the brief is a production spec. The creative team is executing against rules instead of solving a problem.

The 193-page style guide that lived in Sana's office before any AI work began was a prescription document pretending to be a constraint document. Every page told writers what the brand sounded like. Almost no page told writers what the brand had decided about its readers, its claims, or its responsibility to the people it was writing for. The voice could be imitated, badly, by anyone who read the document. The judgment could not, because the judgment was nowhere in writing.

Voice Without a Ventriloquist

In late 2025, I had a long phone call with the editorial director of a health and wellness publisher. Eight writers producing between forty and sixty pieces a week across their site, newsletter, and social channels. They had recently started using AI to generate first drafts, and the results were producing a crisis that looked, on the surface, like a quality problem.

The drafts were fine. Grammatically correct. On topic. Factually accurate, mostly. They did not sound like the brand. The editorial director, a woman I will call Sana, described it this way: "It's like someone studied our back catalog and learned all the words we use without understanding which ones we'd never put next to each other."

Sana had a style guide. 193 pages. It covered everything from comma usage to the preferred way to describe clinical research. It was thorough, internally consistent, and almost useless for the specific problem she now faced. The AI had been given the style guide. The AI was following the style guide. The output still did not sound right.

The first thing her team tried was something almost too simple to count as a method. Sana and her two senior editors sat in a room with a stack of recent AI drafts and a stack of their best human-written pieces and marked, in the margins, every moment where the AI draft went wrong. Not wrong in a grammar sense. Wrong in a voice sense. The moments where a human editor would have flagged something with "this isn't us" without being able to explain exactly why.

They surfaced forty-seven marked passages across twelve pieces.

She and the editors sorted them. Nearly all forty-seven fell into eight categories. The AI was making eight kinds of moves that the brand would never make, and none of these eight were in the style guide because they had never needed to be. The senior editors knew them instinctively. The junior writers had absorbed them over months of getting redlined. Nobody had written them down because, until the AI arrived, nobody had needed to.

Here is the list, which Sana eventually titled "Things We Never Do":

One: We never lead with fear. A piece about sleep deprivation does not open with the health consequences of not sleeping. It opens with the experience of wanting to sleep and not being able to.

Two: We never cite a study without saying what the study actually measured. "Research shows that meditation reduces stress" is not acceptable. "A 2019 trial at Johns Hopkins tracked 142 adults over eight weeks and found that participants who meditated daily reported 23 percent lower scores on a standard stress inventory" is acceptable.

Three: We never use the word "should" in a headline. The brand's position is that the reader is already doing their best and does not need to be told what they should do.

Four: We never write a listicle where the items are interchangeable. If the order does not matter, the piece needs restructuring.

Five: We never quote an expert without including at least one sentence that makes them sound like a person with an opinion, not an oracle dispensing truth.

Six: We never describe a health condition without first acknowledging that the reader might already have it.

Seven: We never close a piece with a call to action that assumes the reader will do something immediately. The closing should leave them thinking, not clicking.

Eight: We never use metaphors drawn from warfare, competition, or conquest to describe the reader's relationship with their own body.

That is the list. It is short. It is specific. It was worth more, operationally, than the 193-page style guide for the purpose of getting the AI to produce usable first drafts.

Constraints defined by negation turned out to be far easier for the system to act on than positive guidance. "Be warm" is something a human can interpret because a human has felt warmth. A language model has not. It produces text that statistically resembles text humans labeled as warm, which is a different thing. "Never lead with fear" is a structural rule the system can check.

The team encoded all eight rules into their AI workflow as a filtering layer. First drafts were generated, then checked against the eight constraints before any human editor saw them. Drafts that violated a constraint were flagged and regenerated. Within three weeks, first-draft usability went from roughly 20 percent to closer to 65 percent. Not because the AI became a better writer. Because the AI was told what the brand had already decided it would never do.

Before / After: creative brief

Before:

"Write a piece about sleep deprivation in a warm, authoritative tone."

After:

"Open with the experience of wanting to sleep and not being able to (never lead with fear-based health consequences). Cite at least one specific study with named institution, sample size, and

> measurement. Close on a thought, not a call to action."

The first brief instructs the writer how to sound. The second tells the writer what the piece has to be true to. The system can verify the second. It cannot verify the first.

The 35 Percent

Around week five, Sana's team started getting drafts through the filter that passed all eight constraints and still weren't right. The system had learned to avoid the eight mistakes. What remained were drafts that were structurally compliant and tonally dead.

I had told Sana the eight constraints would get them most of the way. They got them 65 percent of the way. The remaining 35 percent was a different problem, and I did not have a clean answer. We tried adding positive constraints ("every piece must contain at least one moment of self-deprecation or lightness") and the AI produced mechanical humor bolted onto an otherwise flat paragraph. We tried feeding the system the team's ten best-performing pieces as style examples. The output improved marginally but started to repeat phrasings from those ten pieces, a kind of overfitting.

What eventually helped was a second-pass review by one of Sana's senior editors, working from a short rubric. Does this piece have a point of view. Would a reader be able to tell us apart from a competitor after reading it. Is there at least one sentence I wish I had written. The rubric was not machine-readable. It required a human. The 65 percent held, but the 35 percent got faster to fix because the editors were starting from drafts with no structural problems and could focus on the dimension the system could not capture.

Sana said something during one of our working sessions that I think about often. She had been editing for this brand for six years and had never been able to fully explain to a new hire what the voice was. "The list of eight does in a week what I used to do in four months. And I'm embarrassed it took a machine to force me to write it down."

The knowledge transfer problem she described is universal. It used to be invisible. Every senior creative carries implicit quality standards absorbed through years of practice. These rules are often contradictory when examined closely. Experienced people manage the contradictions through what they call taste, which is the word we use when we mean "judgment I cannot fully explain."

For decades, taste transferred slowly through apprenticeship. A new writer got redlined two hundred times and eventually stopped making the moves that got redlined. The transfer was lossy. It worked well enough, because the volume of creative output was limited by human capacity.

AI broke that equilibrium. When you can generate five hundred pieces of content in a week instead of fifty, the implicit-knowledge transfer model collapses. There is not enough editorial attention to redline five hundred pieces. The taste cannot flow through the bottleneck at that volume.

The Eighth Constraint Was Wrong

I talked to Sana again about three months later. She had added two more items to the list. Before the new tenth, the eighth had to be revised.

Constraint eight, the rule against warfare and competition metaphors when describing the reader's relationship with their own body, had been written with the wellness and mental-health categories in mind. Three months into running the constraints across all editorial output, one of Sana's senior editors flagged a pattern. The constraint was breaking the publisher's fitness and strength-training category, where readers used and expected language like "pushing past your limits," "training to failure," and a vocabulary of effort that was metaphorically adjacent to the warfare frame the constraint had banned. Drafts in that category came back flat and over-soft. Engagement on those pieces dropped by roughly twenty percent over the first eight weeks. The editor argued that the constraint was right for the meditation, sleep, and chronic-illness categories and wrong for fitness, and that bundling them together had been a category error.

Sana rewrote constraint eight to scope it explicitly to mental-health, sleep, and condition-management categories, with a separate posture for fitness content where effort vocabulary was permitted as long as the metaphors did not frame the body as adversarial. The rewrite took a working session and a follow-up. The lesson she took from it was that a constraint that does not name the category it applies to will, given enough volume, get applied where it does not belong, and the system will execute the misapplication faithfully.

The tenth came from a different direction. One of her writers had published a piece about chronic dehydration that opened with the health consequences, bluntly and without softening. It violated constraint one. Sana had approved it anyway, because the writer made a judgment that the topic demanded directness over comfort. The piece performed well. Readers responded to the urgency. An AI following the constraint list would have rejected that opening. The writer overrode the system. She was right to.

So Sana added constraint ten: "We never let constraints one through nine override a writer's conviction that the piece requires an exception, but the exception must be argued in writing before publication, not discovered after." The constraint was about the right to break constraints. I have not seen another team arrive at that move, and I think it is the most sophisticated thing Sana's team produced.

What constraint ten does, structurally, is preserve the part of human judgment that the nine rules cannot encode. A writer with conviction is allowed to break the rule. The rule does not become weaker because of it. The override is documented. The reasoning is on record. If three writers in a year override constraint one for the same reason, that is a signal the rule needs revision, not a sign it was always wrong. The meta-rule turns the rule system from a fence into a learning loop.

I have started suggesting some version of this in every conversation I have about creative intent. Whatever the constraints are, build in the right to break them, build in the requirement to argue the break in writing, and build in the discipline of reviewing the breaks to see which ones should change the rules.

The eight, then ten, did the work of establishing what the brand had decided. The 35 percent that still needed human hands was the work of deciding what the brand had not yet decided. And the exceptions, the moments where a writer overrode the system with a judgment call that turned out to be right, were the edge where the next round of constraints would come from.

That cycle is the job now. Not generating content. The system does that. Not checking compliance. The filter does most of it. The job is operating at the edge of what has been defined, making calls in the territory where no rule exists yet, and deciding which of those calls should become rules for the next cycle.

The most common first reaction to AI-generated creative output, across every creative leader I have spoken with, is not "this is bad." It is "this is close but wrong in a way I can't articulate." That inability to articulate is not a flaw in the person reacting. It is a signal that the standard they are applying was never made explicit.

Before AI, the cost of unwritten creative standards was borne quietly. New hires took longer to ramp. Brand voice drifted slowly across channels. Real costs, spread out and hard to see. After AI, the cost concentrates. A hundred pieces go out with the wrong voice in a single week. The CMO asks why the newsletter sounds different from the website. The cost that was always there becomes visible in a way that forces a response.

Creative work resists the discipline because the output looks like taste, and taste resists being written down. Legal, operations, and finance resist for a different reason. Their output is not taste. It is judgment

that someone will have to defend. That changes the stakes of writing it down, and it changes who the writing belongs to.

8

CHAPTER 8

Legal, Operations, and Finance

When the output is advice, intent architecture is not optional

The cloud infrastructure vendor agreement came back clean. No flags. No escalation triggers. A junior attorney named Mei, on her way out the door on a Friday in October 2025, glanced at the document and noticed a data residency clause that gave the vendor the right to store customer data in any jurisdiction where the vendor maintained infrastructure. She flagged it manually. The general counsel of her 2,000-person insurance company, a woman I will call Priya, reviewed it and was alarmed. Under the company's current understanding of its regulatory obligations, that clause was a serious problem.

Seven months earlier, Priya had told her team that the AI contract review system had paid for itself. Contract review prep had dropped from six hours to ninety minutes. The team was processing vendor agreements faster than at any point in the company's history. The pilot was, by every available metric, a success.

The clean review of the cloud contract was where the success unraveled.

The problem with this kind of work is that the output is not a thing. It is a judgment. When a factory floor speeds up, you get more widgets. When a legal team speeds up, you get more analysis, more recommendations, more advice that other people inside the organization are going to act on. Someone has to own that advice.

A lawyer who reviews a contract and flags a risk is making a judgment. You can ask her why she flagged it. That reasoning has an address. It lives in a person.

A system that flags a contract risk is doing something that looks identical from the outside. The flag appears. The risk is described. The reasoning has no single address. It is distributed across a training set, a set of encoded heuristics, a model architecture that has been updated since deployment, and a human review process that may or may not have examined the specific logic chain that produced this particular flag. You can read the output. You often cannot reconstruct, with any confidence, whose judgment it represents.

The first instinct is to frame this as a legal problem, a liability question about who gets sued when the AI gives bad advice. The framing misses the bigger one. The primary problem is organizational. When a system produces judgment and nobody has decided whose judgment it is, the organization stops knowing what it believes. I heard it happen at Priya's company over five months.

Priya: Whose Judgment Was It?

The setup had been reasonable. The vendor trained the model on a corpus of the company's historical contracts. An outside counsel firm provided heuristics: which clause types to flag, what deviation thresholds to apply, which risk categories warranted automatic escalation.

Here is the sequence of decisions that were never made explicit.

The outside counsel's heuristics reflected that firm's general posture on vendor risk, which leaned conservative. The firm handled litigation for insurance companies and had a natural incentive to flag more, not less. Nobody discussed whether the insurance company's own risk tolerance matched the outside firm's. Priya later told me she assumed someone had calibrated this. Nobody had.

The training corpus included contracts reviewed between 2020 and 2024. During that period, the company had gone through two general counsels with meaningfully different philosophies about acceptable risk in technology vendor agreements. The 2020-2022 contracts reflected a GC who wanted aggressive intellectual property protections. The 2022-2024 contracts reflected Priya, who was more concerned about data privacy and less focused on IP. The model learned from both eras without distinguishing between them. It developed a risk profile that was, in effect, a statistical average of two people who disagreed about what mattered.

The AI vendor updated its base model twice between deployment and the point where problems surfaced. Both updates improved general performance on contract analysis benchmarks. Neither update

was reviewed by Priya or her team. The vendor's changelog was sixteen pages long. Nobody on the legal team had been told that reading it mattered.

For five months, the system ran. Junior attorneys reviewed the flags. They escalated roughly 15 percent of reviews to senior counsel. The numbers looked great. Then Mei caught the data residency clause.

When Priya asked why the system hadn't flagged it, the answer took two weeks to reconstruct. The clause type was in the system's taxonomy. The flag should have fired. The deviation threshold for data residency clauses had been set using the outside counsel's pre-2023 guidance, drafted before a regulatory change that significantly tightened data residency requirements for insurance companies. The outside counsel had updated their own internal guidance in early 2024. They had not updated the heuristics they had provided to the AI vendor. The AI vendor had no mechanism to check whether the heuristics were current. Priya had no process for auditing whether the heuristics reflected her team's current legal position.

Nobody had been negligent, she told me. Everybody had done something reasonable. The outside counsel had assumed the client would update them if priorities changed. The AI vendor had assumed the heuristics they received were current. Priya had assumed the system reflected her team's judgment because her team had been involved in the setup. Her team's involvement had been limited to reviewing a summary of the training process, not auditing the actual decision rules.

The system had been representing a judgment that belonged to no one. It was the ghost of an outside counsel's 2022 risk posture, averaged with two prior general counsels' contract philosophies, running on a model that had been updated twice without legal review. And it had been producing advice, real advice that junior attorneys were acting on, for five months.

I had expected, when she told me, that her anger would land on the vendor or the outside counsel. She was angry at herself. "I should have asked whose view of risk the system was encoding. I didn't ask because I assumed it was mine. It wasn't anyone's."

Priya's fix was slow and unglamorous. Her team spent six weeks producing what they called an analytical charter for the contract review system. Eleven pages. It specified the risk tolerance the system should apply to each contract category, stated as a position the GC's office was willing to defend, not a statistical average of past behavior. It specified the regulatory assumptions the system was operating under, each with a named person responsible for reviewing the assumption on a quarterly cycle. It specified the conditions that should trigger escalation, defined not by risk score but by the presence of clause types where the organization's position was still evolving. It specified a named owner for each

heuristic, a person who could be asked: "Is this still what we believe?" And was expected to have an answer.

The hardest part, Priya said, was not writing the document. The hardest part was discovering that her team had no shared position on several questions she had assumed were settled. The acceptable level of risk in data residency clauses was one. The appropriate response to a vendor's refusal to accept unlimited liability was another. These were questions her senior attorneys answered differently depending on the day, the counterparty, and, she suspected, their mood.

Before / After: legal review

Before (legal prompt):

"Review this contract for risk."

After (intent-driven legal):

"Flag IP indemnity caps below $5M. Flag data residency clauses that allow processing outside [list of approved jurisdictions]. Flag unlimited liability waivers. Auto-escalate any clause type the team has not reviewed in the last 90 days. All flag thresholds owned by GC, reviewed quarterly."

The prompt asks for an opinion. The intent statement names whose opinion, on what, against which thresholds, valid until when.

Operations: A Document That Failed by Following Itself

A COO at a consumer products company told me, six months into her attempt to write an intent document for her function, that the work had felt like being asked to predict the future. She was right that it felt that way. She was wrong that it actually required it.

Her function had been running on rules of thumb. How much inventory to hold for which SKUs. When to expedite a shipment versus accept the delay. Which vendor disputes to escalate to legal and which to resolve at the buyer level. None of it had been written down. When the company brought in a planning system with AI-driven forecasting and replenishment, the system needed instructions she had never given anyone in explicit form.

I told her, on the call, that she did not have to convert all of her judgment. She had to convert the parts the system needed to act on without her in the room. The rest she could keep. The first version of her intent document, after four weeks of work, was four sentences. Top 20 percent of SKUs by revenue: 14-day buffer above forecast demand. Next 30 percent: 7 days. Bottom 50 percent: accept stockout risk and replenish on demand. Revisit if vendor lead times exceed 21 days or demand projections shift more than 15 percent in either direction. Escalate any single-vendor dependency representing more than 8 percent of category cost of goods.

That captured enough for the planning system to act on inventory decisions without paging her, and to flag the cases that fell outside the rules for her actual attention.

Five months later, the document produced the kind of failure that exposes how rules behave in the wild. The company launched a seasonal SKU, a holiday gift bundle, that the planning system classified into the top-20-percent revenue band based on early sell-through. The system applied the 14-day buffer policy and placed replenishment orders against it. Three weeks before the peak demand window, the buffer ran out. Stockouts cost the company an estimated $340,000 in lost sales over twelve days.

The cause was not the policy. The policy had executed correctly. The cause was that the policy had been written for steady-state SKUs whose demand patterns held across weeks, and seasonal bundles had a different shape. The 14-day buffer was the wrong instrument for a SKU whose demand curve had a vertical front edge.

A regional planner had flagged the bundle two weeks before the stockout and recommended a 30-day buffer. The system did not surface his recommendation because his override was outside the document's rules. He was correct. The document was incomplete. The COO amended it that quarter to add a seasonal-SKU category with a different buffer formula and a named escalation path for any item launched with a projected six-week revenue window or less. She told me, with some heat, that the post-mortem was harder than the launch had been, because the document had been clean enough for everyone to follow and incomplete enough to fail expensively while looking like it was working.

She still makes the exception calls. The system flags candidates. She decides which ones are real exceptions and which are signals to update the rule. The exceptions she handles personally have dropped from roughly forty a week to twelve, and the twelve are the ones that actually need her.

Finance: An Override That Exposed a Missing Rule

The CFO at a logistics SaaS company, a man named Marcus, told me in early 2026 that he had been the bottleneck on revenue recognition for three years and hadn't seen it. Every deal that involved a custom commercial structure, multi-year prepay with delivery milestones, partial revenue acceleration for early commit, contingent professional services, came to him for review. The finance team did not feel competent to make the call. The deal desk did not want to be wrong. So Marcus reviewed each one, often by Slack, often in twenty-minute windows between other meetings.

The company closed about 180 deals a year that needed his judgment on rev rec. Roughly four a week. His most senior person was spending the equivalent of one full day a week on judgments that, individually, were not actually that hard.

He started the intent work after a deal got booked under a revenue treatment he later disagreed with. The deal desk had asked, Marcus had responded with a thumbs-up emoji at 11 p.m. without reading the structure carefully, and the auditors had questioned the recognition six months later. No restatement, but uncomfortable.

He wrote a four-page revenue recognition policy specific enough for the deal desk to apply without his involvement on most deals. Multi-year prepay with delivery milestones: recognize ratably over the contract term unless a single milestone represents more than 40 percent of contract value. Partial acceleration for early commit: allowed up to 8 percent of contract value, capped at $200K per deal, treated as discount rather than acceleration. Contingent professional services: recognize on delivery only, never on contract signature, regardless of customer accounting treatment. He added matching policies for capex thresholds and accruals.

The deal desk started handling rev rec calls without Marcus on roughly 70 percent of deals within two months. The other 30 percent still came to him, but they were the deals where the policy genuinely did not cover the structure, which were the deals that deserved his attention. The auditors were happier. He stopped getting Slack messages at 11 p.m.

Eight months in, the policy got tested. A senior AE pushed back on a rev rec determination the deal desk had made on a large strategic account, a three-year contract with a customer-funded development component that the policy classified as contingent professional services. The deal desk had applied the rule cleanly. The AE argued the development work was structurally different from a normal professional services engagement because the customer was paying for outcomes that would become part of the shipped product, available to other customers afterward. He wanted the development revenue recognized on delivery of the underlying product features, not on the services delivery schedule.

Marcus reviewed it and agreed with the AE. The deal desk had applied the rule correctly. The rule had not anticipated a category that did not exist when he wrote the policy. He amended the policy to introduce customer-funded development as a distinct treatment category with its own recognition logic, and he wrote into the next version of the document a question for himself. Which other categories had he failed to anticipate that the deal desk was now classifying into the closest available bucket and getting subtly wrong. He found two more over the next quarter. The policy grew from four pages to six.

He told me the AI was not actually doing the rev rec calls. The AI just sat there as a structured system waiting for instructions, and he could not pretend, with a system in the loop, that the policy lived in his head. The system forced him to externalize what should have been written down years earlier. The forcing function was not the AI's competence. It was the AI's incompleteness.

The Ownership Question

There is a move that nearly every organization makes when the accountability problem surfaces. They add a human review step. A lawyer signs off on the AI's output. A controller reviews the rev rec treatment. An ops manager approves the buffer recommendation. This feels like it solves the problem. A human is in the loop.

It does not solve the problem. It relocates it.

When a human reviews AI-generated analysis, what are they reviewing against? If the answer is "their own judgment," the system has not added anything. You have an expensive first draft and a human doing the same work they were doing before. If the answer is "the organization's standards," then the question becomes where are those standards written down, do they reflect current positions, has anyone checked recently.

Priya's junior attorneys were the human in the loop. The system still produced advice that reflected a judgment no one owned, because the review step checked whether the flags were reasonable, not whether the flagging criteria represented the organization's current legal position. The review was a quality check on the output. It was not an ownership check on the reasoning.

The alternative is prior commitment. Before the system produces output, the organization decides what it believes about the questions the system will be answering, whose interpretation is authoritative, what conditions would change the answers, who is responsible for noticing when those conditions have changed.

Priya's charter. The COO's buffer policy. Marcus's rev rec rules. Each one answers the question the system cannot answer for itself: according to whom.

The asking turns out to be worth more than the document it produces. The document will be revised, debated, and probably rewritten within a year. The asking forces people who have been quietly disagreeing to disagree in the open. It forces leaders who have been delegating judgment to claim it.

In a function where the output is judgment, you cannot have unowned judgment running for five months without consequences. The intent document is the answer to "according to whom." Without it, the answer is "no one in particular," a position no functional leader actually wants to take but many take by default because writing down the alternative felt harder than it was.

Revenue exposed the cost of vague intent at the email level. Creative exposed it at the voice level. Legal, operations, and finance expose it at the ownership level. Product and strategy expose it somewhere stranger. The question is no longer whose reasoning the system represents. The question is what the organization actually wants. And the organization, very often, has not decided.

9

CHAPTER 9

Product and Strategy

The roadmap as an intent system

Two AI-generated research briefs, produced by the same consulting firm for different clients in the same industry, reached contradictory conclusions about the viability of a specific distribution channel in Indonesia. Both briefs had been reviewed by senior analysts. Both had been sent to clients. Neither analyst flagged the contradiction because neither had seen the other brief.

The partner who noticed, a man named Eli, raised it in a partners' meeting in early 2026. Three of the six partners disagreed about the right answer. The disagreement was not new. It had existed for at least two years. It had never created a visible problem because, in the old workflow, a partner would personally review any recommendation that touched the topic and apply their own judgment. The client got that specific partner's view, not the firm's view. Nobody had thought of this as a problem because nobody had thought of the firm as having a view distinct from its individual partners.

The AI did not have a partner's judgment. It had a training set built from the firm's past deliverables, which contained both sides of the disagreement. It produced whichever conclusion the data in its context window supported more strongly, which varied by client.

Eli's contradiction is the same problem as Priya's data residency clause, scaled up. Priya's system reflected risk positions nobody owned. Eli's firm produced recommendations nobody had agreed to.

Both are forms of strategic ambiguity made visible by a machine that cannot hold ambiguity the way humans can.

Eli's case is worth dwelling on because it surfaced three long-held methodological disagreements between senior partners that had been quietly coexisting for years. The first was the distribution channel question itself. Two partners believed direct-to-consumer digital channels were viable in most Southeast Asian markets for consumer goods priced under fifteen dollars. One partner, a woman named Rena, believed offline distribution through traditional trade remained dominant for that price point and that recommending digital-first was irresponsible. She had data. The other two had data. The data pointed in different directions depending on the product category and the country.

The second disagreement was about how to weight government regulatory risk in market entry recommendations. Eli treated it as a near-veto factor. Another partner treated it as a background condition to be managed. Their past deliverables reflected both approaches, interleaved across years of client work.

The third was the most interesting. A disagreement about what counted as a primary source. Two partners required all strategic recommendations to be grounded in proprietary survey data or on-the-ground interviews. Rena and one other partner accepted published industry reports and government statistics as sufficient primary sources for certain types of analysis. This had produced, over the years, deliverables of visibly different evidentiary rigor, a fact nobody had named but that the junior analysts knew well. ("You always over-research for Eli's clients and under-research for Rena's" is how one analyst described it to me, off the record, with a slightly guilty expression.)

None of these disagreements were secret. Every partner knew the others' positions. They had simply never needed to resolve them, because the humans in the workflow could adapt in real time. When Eli reviewed a brief, his priorities shaped the output. When Rena reviewed a brief, hers did. The client got a consistent product because one person's judgment governed each engagement. The AI could not do this. It produced output that reflected an averaged, unresolved version of all six partners' views, which is to say it reflected no one's view.

This is what makes product and strategy work different. The question is no longer "whose judgment does the system represent." The question is "what does the organization actually want." And in many organizations, the answer to that question has been deliberately deferred for years, because deferring it kept the coalition intact.

The Roadmap Is an Intent System

Most roadmaps say what will be built. They do not say what is being optimized for, what trade-offs have been decided, what would change the priority order. They are output lists with quarterly tags.

A roadmap that says "Q3: improve onboarding" is doing the same thing the payments company's old ICP document was doing. It is a phrase vague enough to contain multiple incompatible interpretations, which is convenient for getting the document approved and useless for telling a system, or a team, what to actually do.

I have started asking product leaders, in the first conversation, to take a single line item from their current roadmap and tell me three things about it. What is the metric this is supposed to move. What is the trade-off the team has accepted to prioritize this. What would cause us to reconsider. Almost nobody can answer all three in the first attempt. Most cannot answer the second. The trade-off question is the one the roadmap has been written specifically to obscure.

Before / After: roadmap line

Before (roadmap line):

"Q3: improve onboarding."

After (intent-driven):

"Q3: cut day-7 churn for users acquired through paid channels by 30 percent (currently 41 percent). Trade-off: deprioritize power-user features. Open: whether to extend the change to organic acquisition. Revisit if paid CAC exceeds $X or organic mix climbs above 60 percent."

The first version is a banner. The second is a decision the team can defend, contest, or override. The second also forces the disagreement that the first was designed to avoid.

Two PMs, Two Target Users, and the Third That Was Missing

In the fall of 2024, a head of product at a Series C software company told me about a tension brewing on his team. Twenty-six engineers. Four PMs. The company was at the point where the roadmap could no longer be held in one person's head, and he had started writing intent documents for the next two quarters' work.

Two of the PMs, one running the platform team and one running the growth team, both believed they were building for the same target user. The phrase they used was "the user." When pressed, the platform PM described someone who had been using the product for at least six months and had three or more integrations configured. The growth PM described someone who had signed up in the last thirty days and had not yet completed onboarding.

These are different people. The product changes that helped one would hurt the other. The decisions about API limits, default settings, in-app prompts, navigation hierarchy, all of them were being made by two PMs working from incompatible mental models of who the product was for.

The intent document forced it open. When the head of product asked each PM to write down, in one paragraph, the user the next quarter's work was for, the paragraphs did not match. The disagreement that had been invisible in the roadmap became impossible to ignore in the intent document, because the intent document required specificity that the roadmap did not.

The conversation that followed was painful. The growth PM argued that new users were the lifeblood of the business and that prioritizing power users was investing in a smaller, slower segment. The platform PM argued that power users drove expansion revenue and that pivoting to new-user features would degrade the experience for the customers actually paying. They were both right. They could not both be right at the same time, in the same quarter, with the same engineering capacity.

The head of product made the call. The next two quarters would prioritize new-user activation, with explicit deprioritization of power-user feature requests below a defined revenue threshold. The platform PM was unhappy. He said so. The decision was logged in the intent document along with the trade-off and the revisit condition.

Six months later, the company had cut day-7 churn meaningfully and had lost a small but real number of expansion deals from power users who felt deprioritized. Both things were true at once.

There was a third thing the document had not anticipated, and the cost of that omission only surfaced ten months in. The intent document treated the company's customers as two segments, new users and power users, and built the trade-off between them. It did not name a third segment, which the head of product later described to me as embedded users: end users at large enterprise accounts who did not sign up themselves and were never onboarded through the funnel the growth PM was optimizing, but whose product usage drove renewal decisions made by an entirely different procurement contact. This segment had been roughly twelve percent of monthly active users and, by the time anyone looked closely, was generating about thirty-eight percent of annual contract value. The activation-focused work the team

had shipped for ten months had been small to neutral for this segment, and a few of the navigation changes had been actively worse for them. The cost surfaced when a $1.8M renewal nearly walked over a workflow regression in a feature embedded users relied on.

The head of product told me, when we talked about it later, that he had read the intent document literally fifty times during those ten months and never once noticed that it described the customers in terms that did not include the most valuable ones. The document had described what the team was deciding between. It had not described what the team had failed to put on the table. He added a third segment for the next quarter and a question, written at the top of the document in italics, that he intended to keep there permanently. Are these the segments, or are these the segments we are currently arguing about. The latter is a smaller set than the former, almost always.

He said the original decision had been the right one for the company at that stage and the wrong one for some specific customers, and that he was at peace with that because the intent document had made it possible to have the argument out loud. The third-segment miss, he said, was the part he was not at peace with, because it was the part the document had been written to prevent.

Strategic Ambiguity Is Often Load-Bearing

Strategic ambiguity is often deliberate. Coalitions hold together because no one is forced to choose. The roadmap that says "improve onboarding" survives the planning meeting because the platform faction reads it as "improve onboarding for power users" and the growth faction reads it as "improve onboarding for new users" and both factions vote yes. The ambiguity is not a failure of writing. It is a feature of governance.

This is why intent architecture is harder in product and strategy than in revenue or creative. In revenue, the intent document mostly clarifies what was already known but unwritten. In product, it forces the organization to make decisions it has been actively avoiding. The disagreement is not an artifact of poor communication. The disagreement is the actual disagreement, and it has been managed for years by leaving the language soft enough to contain both sides.

AI makes that strategy untenable, because the system cannot hold the ambiguity. It either prioritizes power-user features or it doesn't. It either weights new-user activation or it doesn't. It cannot do the human thing of treating both as equally important on Monday and quietly doing one of them on Thursday.

I went into my first three conversations with product leaders on this thinking writing intent documents would clean up roadmaps. For all three, the documents had made the conflict worse before they made anything better. The product organizations had been running on a kind of distributed denial that allowed factions to coexist without resolving their disagreements. When the intent document demanded clarity, the denial collapsed. People who had been working together amicably for years discovered they had been disagreeing the whole time. The months between visibility and resolution had been brutal.

I now warn product leaders about this when they ask. If your team has been running on strategic ambiguity, the intent exercise will not be clean. It will surface conflicts you have been managing for years, and you will have to actually resolve them, and some of your team will be angry about decisions they were quietly relying on you to never make. That is the cost. It is also the point.

What the System Cannot Hold

The thing AI cannot do, in product and strategy, is the thing that holds many organizations together. It cannot manage productive ambiguity. It cannot read the room and quietly route around the disagreement. It cannot sit in a planning meeting and notice that two leaders are using the same phrase to mean different things and decide that today is not the day to surface that.

A human leader does this kind of work constantly, often without knowing they are doing it. It is part of what experienced operators get paid for.

The intent document refuses that compromise. It asks for the trade-off in writing. It asks for the revisit condition in writing. It asks what would change the priority order, which is a question many leaders have been carefully not asking themselves.

After the exercise, a head of product told me: "I'd been keeping my options open. I didn't realize how much of my authority came from being slightly hard to pin down."

That is the trade. The intent document gives the organization clarity. It costs the leader some of the flexibility that comes from being interpretable in multiple ways. Most leaders, when they understand the trade, choose clarity. Some don't. The ones who don't usually find that the AI rollout stalls without obvious cause, and the cause is usually them.

The Eli partners eventually wrote position papers on the three disagreements. They did not reach consensus on all three. On the distribution channel question, they wrote a paper that said, in effect, "We believe digital-first is viable for these specific product categories in these specific markets, and we believe traditional trade remains dominant for these others, and where the evidence is genuinely

ambiguous we will say so to the client rather than defaulting to either position." On the regulatory risk question, they agreed on a weighting approach. On the primary source question, they are still arguing, and Eli told me he suspects they will never fully agree.

That is fine. The intent document does not require agreement on everything. It requires clarity about where agreement exists, where it does not, and what the system should do in the zone of disagreement. For the research team, the answer in the disagreement zone was: flag the output for partner review and include a note identifying which analytical assumptions drove the conclusion. Not a perfect solution. A real one.

I went into that conversation thinking the methodological disagreements were a problem to be solved. I now think they were a resource the firm had been accidentally wasting. The disagreements represented genuine intellectual diversity about hard questions. The old workflow hid that diversity from clients. The intent architecture exercise made it visible, and several clients told Eli they preferred the new approach, where the brief explicitly stated which assumptions were contested and what the analysis looked like under each set of assumptions, to the old approach where they received a single recommendation and assumed it was the firm's unified view.

Forward

The four functions in this part, revenue, creative, legal and operations and finance, product and strategy, are not separate cases of the same lesson. They are four pressure points on a single problem. Each function exposes a different version of the question the system is trying to answer for the organization, and the answer in every case requires the organization to externalize judgment that had been living implicitly inside specific people.

The shape of the problem changes with the function. In revenue, the implicit thing is the operating definition of a good lead. In creative, it is the negative space of what the brand will not do. In legal and operations and finance, it is the position the function is willing to defend in writing. In product and strategy, it is the trade-off that has been kept ambiguous to hold the coalition together. Each of these can be written down. Each costs something to write down. Each pays back more than it costs, eventually, though rarely on the timeline the leader expected.

If the discipline can be applied across every function, then the operating model itself has to change. The intent document is not a tool that bolts onto the current operating model. It is the beginning of a different one, in which the organization's beliefs are written down and the system runs on top of them. That has consequences for how teams are structured, how decisions get made, what work managers actually do,

and what the organization is for.

That is the next part.

10

CHAPTER 10

The Intent-Driven Organization

What it looks like when a company has actually written down what it wants

Priya's weekly leadership meeting got forty percent shorter.

That is the number she gave me four months after her team finished the analytical charter. Not a few minutes. Forty percent. I asked her why and she said the arguments had changed. Before the charter, her senior attorneys spent meeting time relitigating specific contracts, and those debates were proxies for a risk-philosophy argument nobody wanted to have. After the charter, the risk philosophy was on the page. The remaining arguments were about whether the page was still right, or about cases that fell outside the page's scope. Cleaner argument. Faster meeting. Same eight people.

I have heard a version of that finding from eleven of the eighteen leaders who told me about going through this kind of work between late 2024 and early 2026. The meeting got shorter. Not because AI did the talking. Because the company had finally written down what it believed.

That is what an intent-driven organization is. Not more automation. Not a control room. A company that has paid the price of clarity in advance, in writing, so machines can run against that clarity at speed.

The villain of this book is implicit intent at machine speed: a system producing thousands of outputs per day against assumptions nobody ever wrote down. At scale, the result is a well-organized time capsule. The system efficiently reflects whatever the organization happened to believe the last time anyone

bothered to check, which was usually some Tuesday in 2025. The opposite is not better prompts. It is the company in this chapter.

Monday Morning in an Intent-Driven Company

A Monday in an intent-driven company looks duller than the brochure version. No glowing dashboards. No control room. The work is mostly reading.

Picture the head of marketing at a B2B SaaS company that has done this work. Call her Marisol. She starts her Monday by opening the brand intent document. Version 4.7. The last change, two weeks ago, added a constraint about how the brand handles competitor comparisons, after a piece of content went out that made her uncomfortable. She reads the document. She reads the override log from the last seven days. She looks at the drift report.

The drift report is unfamiliar to most leaders at first. Not a performance dashboard. A record of where the system's output has been moving away from what the intent document specifies. Not errors. Drift. Across two hundred drafts produced last week, the AI started using a slightly more formal tone than the document calls for, because the model has been shifting since the last vendor update. Not flagged anywhere else. The drift report is the only artifact that catches it.

She makes a note. She does not call a meeting.

A few floors down, a senior analyst named Reza is looking at his queue. The system has flagged three pieces of work as falling outside what the intent document covers. A partnership tier that did not exist at the last revision. A customer escalation in a region the document says nothing about. A case where the constraints conflict. Reza spends about thirty minutes on each. The resolutions go into the override log.

Trade-offs are visible. The marketing document includes a section called "What We Do Not Optimize For," which lists the two things the brand is willing to lose to keep what it cares about. The legal team's analytical charter says, explicitly, that the firm will accept a higher false-positive rate on data residency flags in exchange for a lower false-negative rate, and assigns a partner to revisit that ratio if the cost of investigation crosses a stated threshold. Unsexy sentences. Load-bearing wall.

Strategy is versioned. The documents have version numbers and changelogs. A new hire does not learn the brand by sitting next to a senior person for six months. She reads the document, the last three months

of override decisions, and the open questions section, where the team has agreed they have not yet agreed. Two weeks instead of two quarters.

The AI systems read the document before acting. That sentence sounds science-fictional. It is plumbing. The document is loaded into context through what the engineering team calls an execution hook: the point in a workflow where the system reads the intent document and acts on its contents before it produces output. The routing rule for inbound leads consults the segment priorities at the moment of routing. The forecasting layer reads the trade-off register before projecting next quarter's numbers. The contract review system pulls thresholds from the analytical charter at the moment of review. Without execution hooks, the document is a slide deck. With them, it is the operating instruction set.

When the system is uncertain how a new section interacts with an existing one, it flags the ambiguity before execution instead of guessing. The flag goes to the document owner, not to Reza. Ambiguity routes upstream, before output.

Every override is a learning signal. Each correction Reza makes goes into the log with the reason. Weekly, the document owner reads the log and decides which overrides are one-offs, which are patterns the system should learn from, and which mean the document needs a new section. The governance loop is a weekly read of a log, not a steering committee.

Every section has an owner, a revisit trigger, and an expiration condition. Data residency is owned by a specific partner and revisits when EU regulation changes or quarterly. Brand voice is owned by Marisol and revisits when share of voice in a defined competitor set moves by more than a stated amount. The system does not own anything. People do.

A CRO I spoke with last year told me he stopped opening pipeline conversion rates as his first action on Monday. He reads the drift report against the qualification criteria he wrote. Pipeline numbers tell him what happened. The drift report tells him whether the system's reading of his strategy is still what he meant. The dashboard is downstream. The drift report is the steering wheel.

Institutional memory has been externalized. Some of it badly, in places where the writing is rough. Externalized, which means transferable. Reza went on vacation last month. His seven-step prompt chain did not go with him, because it has dissolved into the document and the hooks. A different analyst can run the system against the same intent and get output recognizably from the same company.

That is Monday. No demos. Nobody amazed. The work is quiet.

What Disappeared

The most visible feature is what is no longer happening. The daily translation work is gone, mostly. The prompt chains. The "can you take a quick look at this" emails. The version of Derek's morning where he constructs a seven-step process to produce output he needed last week and the week before that. That whole shape of work has lifted out. In its place is something more durable: a document the system reads, an override log the owner reads, and a drift report a few senior people read on Monday. The translation tax has been paid once, at the document layer, instead of a thousand times a week at the keyboard.

Prompting still happens, but at the edges. When a new product launches and the existing categories do not yet apply. When a regulatory change opens a section of the document that has no current owner. When Marisol adds constraint nine and constraint ten three months after writing the original eight, because the system's output has surfaced a pattern she had not previously articulated. The volume drops by about an order of magnitude.

I used to think the intent-driven organization was defined by what the technology could do. I have changed my mind. It is defined by how much the organization has been willing to commit to paper before the technology runs. Two companies on the same vendor, the same model, the same price, will produce wildly different output if one has done the intent work and the other is still running on individual prompting skill. Same software. One team gets output it can use. The other gets output that sounds right and says nothing.

The VP of Marketing and the Head of AI

I will talk about two roles and skip the rest. The original sketch of this argument went role by role, and the parallelism made every change sound equally clean. None of them are. Two are interesting. The rest are contingent on the specific organization.

The VP of marketing in a prompt-and-response company spends a meaningful part of her week reviewing AI output. Reading drafts. Catching tone mismatches. Sending things back with notes that say "closer but not quite." She is the quality filter, because the system does not know what quality means for her brand. Her calendar is full of review meetings. Her inbox is full of variations on "can you take a

quick look."

The same role in an intent-driven company does not do that work. The constraints are in the document. The filtering happens before she sees anything. What reaches her is output the system is confident about (which she spot-checks) or output the system has flagged as uncertain.

Her week looks different. She spends more time on the questions that produce constraints and less time enforcing them. Does the existing voice still hold for the new channel we are entering next quarter. A competitor just shifted the category conversation, so does our positioning still work. A piece of content went out last week that followed all ten constraints and still felt wrong, and the question is why.

That last question cannot be automated. The VP's job has shifted from quality control to judgment about things the system cannot yet handle. For some VPs this is a better job. For others it is disorienting. The review work was tedious and legible. You knew when you were done. The new work does not have a clean stopping point. Two strong marketing leaders I have talked with, excellent at the old version of the job, told me they were struggling with the new one because nothing on their calendar told them they were finished for the day. One of them asked, half joking, whether I could put back the "can you take a quick look" emails. She said she missed the small wins.

The Head of AI's role changes in a direction that catches almost everyone off guard. In a first-generation deployment, the Head of AI is a deployment officer. Picks vendors. Manages integrations. Runs pilots. Reports metrics. In an intent-driven company, that work does not disappear, but it becomes a smaller fraction of her time. The larger fraction is organizational translation: working with function leaders to turn implicit beliefs into written intent documents, then maintaining those documents as the organization's positions shift.

Most technologists did not sign up for this, and some are not well suited to it. A Head of AI at a consumer goods company I spoke with in late 2025 described the meeting that finally pushed her toward the door. The CMO had called a session to decide whether the brand's tone document should permit exclamation points in push notifications, and in which categories. The conversation ran ninety minutes. The CMO's deputy thought exclamation points read as friendly on promotional pushes but aggressive on shipping updates. The head of social believed they were fine across the board, but should disappear from any push that mentioned a price discount. The brand director wanted the rule pinned to time of day. The Head of AI sat there and realized nothing in her computer science training, her four years at the previous AI vendor, or her PhD had prepared her to mediate a fight about punctuation.

"I went to school for computer science," she told me. "Now I'm mediating arguments about whether we use exclamation points in push notifications." She had taken the job expecting to run model evaluations, pick integrations, get infrastructure right. What she had been doing for six months was sitting in rooms with marketing leaders, asking them what they meant by "on-brand," watching them realize they did not agree, and writing down what they finally decided in a form the system could read. She quit six months after she described that meeting to me.

Her replacement came from the brand strategy team with no engineering background. The replacement spent her first week sitting through the same kind of meeting and recognized it instantly: a cross-functional brand argument she had moderated her entire career, except now there was a system in the room that needed the answer in writing. She knew how to break the argument open by asking the right next question. She knew when "we'll figure it out as we go" meant the senior people had not yet agreed. She kept a Notion page tracking which constraints had been re-argued more than twice without resolution, and walked into the next session with that page in hand. The constraint documents she produced in the first quarter were longer, more specific, and got argued over less. Output quality on the marketing side moved noticeably within two months.

This is the pattern, not an anomaly. The AI title is attracting people for one job and turning into another, and the gap between the posting and the work is wide enough that the better hire for a Head of AI in 2026 is often someone who has never written a line of code.

I am not arguing every Head of AI should be replaced by a brand strategist. I am saying the job description has split. The deployment half can stay technical. The intent-maintenance half is a different person, often with a different background. Companies that try to keep the role unified often lose the person who took it for the engineering reasons, because the engineering reasons have become a smaller fraction of the day.

Strategic Ambiguity Becomes a First-Order Cost

A finding I did not expect: of those eighteen leaders, fourteen said the hardest part of intent architecture was not the AI work. It was getting leadership to state, in clear language, what the organization's actual positions were.

Most companies do not know what they want with the precision a system requires. Priya did not know whether her firm had a position on data residency risk. Eli's partners did not know whether the firm believed in digital-first distribution or traditional trade for sub-fifteen-dollar consumer goods in Southeast Asia. Sana did not know, until the AI forced it, that her editorial voice had never been fully defined. The work was strong because the humans in the workflow could paper over the ambiguity in real time, never needing to resolve the underlying disagreement because the disagreement never surfaced in a way that demanded resolution.

The system demands resolution on the questions it is going to act on. That demand has surfaced something always true but never costly enough to confront: strategic ambiguity is not a flaw in how organizations communicate strategy. It is a feature. Most companies are strategically ambiguous on purpose, because ambiguity is how coalitions hold together. A strategy document that says "we will grow in Southeast Asia" keeps everyone aligned because the partner who believes in digital-first and the partner who believes in traditional trade can both see their position reflected. The moment you write an intent document that says "our system should recommend digital-first distribution for products under fifteen dollars in Indonesia and Thailand, and traditional trade for products under fifteen dollars in Myanmar and Cambodia," you have made the decision the strategy document was designed to avoid making.

Implicit intent at machine speed produces a coherent-looking output stream that is, on inspection, a thousand small bets the organization never explicitly made. Explicit, maintained intent is the only thing that survives at that speed. Not because the document is perfect. Because the document is visible, owned, and changeable, which means the bets can be corrected when they are wrong.

What This Organization Costs

A version of this chapter sounds like a pitch for fewer people. Systems that know what they want without being asked. Documents that replace individual judgment. Output that flows without prompting. If you are reading this as an argument for cutting headcount and letting the machines run, I have failed to make myself clear.

The intent-driven organization is more dependent on human judgment at the strategic layer than any organization that came before it.

In a prompt-and-response company, the judgment cost is distributed. Everyone spends a little of their day translating for the system. Low-level judgment applied constantly: is this good enough, does this match our voice, should this be escalated. Real judgment, spread thin.

In an intent-driven company, that distributed cost goes away. The system handles it. What remains is the judgment that cannot be pre-answered. What do we believe about data residency given Q3 regulatory changes. Should we break constraint number one because this specific piece of content in this specific context is the exception that proves the constraint was too rigid.

That last one keeps me honest about the limits of the approach. Sana told me her team's best-performing piece of content in Q1 2026 violated constraint number one, the "no fear-based leads" rule. A human made that call. The system could not have, because the system's job is to follow the constraints, not to know when the constraints are wrong.

The intent-driven organization needs fewer people doing translation and more people doing the judgment the document cannot pre-answer. Whether that nets to fewer people overall depends on the company, the function, and what the former translators can do instead. I have heard about teams shrinking. I have heard about teams staying the same size with different work. One leader described a team that grew, because the intent exercise had revealed the company had been under-investing in the senior judgment the new model depended on.

The next chapter is about the most expensive failure mode I have seen, which is almost always made by leaders who think the new operating model gives them their afternoons back.

11

CHAPTER 11

The Leader's New Job

From approving output to designing the conditions for it

A forty-five-minute hole.

That is what the CRO at a logistics company in Chicago found on his Wednesday calendar in March 2026, where his weekly pipeline review used to live.

The meeting had run ninety minutes for as long as anyone remembered. Twelve reps walking through their top deals. He would ask the questions he always asked: what is the next step, what is the decision criteria, what is the risk. He would push on the soft ones. He would close the meeting feeling like he had done his job.

Three months earlier, his team had stood up an intent-driven workflow against his stated qualification criteria. The system did the first pass. By Tuesday night, it had already flagged the deals that did not meet his criteria, surfaced the ones it was uncertain about, and produced a summary of the rest. By Wednesday morning, the meeting was half as long, because the questions he used to ask had been answered before he walked in.

He told me later he did not know what to do with the forty-five minutes. He had assumed he would feel relief. He had felt something closer to vertigo. He said it like this: "What is my job now?"

That is the question. I have heard some version of it from every operating leader who has talked to me about this transition. The honest answer is that the job has changed in a way most of them did not sign up for. The leader who used to approve work now designs the conditions under which work happens. The leader who used to be the final reviewer becomes the editor of the document the reviewer is reading from. The artifact has moved upstream, and the leader has to move with it.

You can refuse the move. Plenty of leaders do. They keep the ninety-minute pipeline review because they like it, or because they are not sure what the new shape is, or because the old meeting was a place they felt useful. The system runs underneath them and the meeting becomes ceremonial. Some of these leaders are gone within a year, replaced by people willing to occupy the new shape. Some find a smaller version of the old role to grow into and stay. The CRO in Chicago was one of those. He found his job again, he told me. It took him about six months. He was still figuring out the edges.

Editor of Intent, Not Approver of Output

The single biggest change is the move from approving to editing.

In the old model, the leader sat at the end of the pipeline. Work came up. Decisions came up. The leader looked at the work, applied judgment, and either approved it, sent it back, or made the call directly. The judgment was applied late, at the point of execution, on a per-item basis. A VP could spend her whole week applying judgment to a hundred items and never write any of it down.

In the new model, the leader sits upstream. The document is the work. Every piece of judgment that used to be applied late, at the point of execution, has to be applied early, at the point of writing. Not perfectly. Not all at once. But explicitly, in a form a system and a junior team member can both read.

That is what I mean by editor of intent. The leader is no longer approving the thousand outputs the system produces. The leader is editing the document the system produces them from. When the brand voice constraint is wrong, the fix is not to send back the latest piece of content. The fix is to change the constraint, watch the next hundred pieces, and see whether the new constraint produces what was actually wanted.

That is a slower, scarier loop. The feedback comes from the aggregate, not the individual. The leader cannot tell, from one piece of content, whether the constraint is wrong. She can tell, from twenty pieces and the override log, that the constraint is producing a pattern she does not want, and that the document

needs a new section.

Most leaders find the document work less satisfying than the approval work. Approval is concrete. You see the piece, you make the call, you move on. Document work is abstract. You write a sentence, you wait, you read a log, you write a different sentence. The dopamine cycle is longer and the win is harder to point at.

I had this wrong for a while. I told leaders, in 2025, that the new model would free them up. I said they would get their afternoons back. I was wrong, and I was wrong in a specific way: the leader's calendar has fewer review meetings, yes, but the time gets eaten by conversations about what the team actually believes, the ones leaders have described to me. Those conversations are harder than the review meetings ever were. The review meeting had a finite agenda and a clear endpoint. The conversation about what the team believes has neither. It often ends with the leader and one or two senior people staring at a sentence and disagreeing about whether it says what they meant.

I stopped promising leaders their afternoons.

The Four Parts of the New Job

The new job has four parts. Each is shaped differently.

The first part is writing and maintaining intent. The document and its trade-offs, including the explicit "What We Do Not Optimize For" section that the original strategy documents almost never contained. The leader is responsible for the document existing, being current, being read, and being honest. Honest means it says the things the leader actually believes, not the things the leader would say in a board meeting. There is a difference. A useful intent document captures the disagreements inside the team, not just the consensus.

Maintenance is harder than writing. The first version gets written in a workshop, with energy, with a deadline, with the satisfaction of a thing produced. The second version, four months later, after the override log has gotten interesting, has to be written on a regular Tuesday, in the middle of three other priorities. Many leaders write the first version and never write the second. The document goes stale within a quarter. The villain returns through the back door.

The second part is reviewing drift. Not approving output. Not reading dashboards. Reading the report that shows where the system's behavior has been moving away from the document. The CRO in Chicago told me he started looking at his drift report on Monday mornings. The first thing he noticed was that the system had been gradually deprioritizing deals from one industry vertical, because the qualification criteria he had written six months earlier had a sentence that, on inspection, biased against that vertical in a way he had not intended. The drift report was the only place that pattern was visible. The pipeline numbers showed the outcome. The drift report showed the cause.

Reviewing drift is the part most leaders skip the longest, because it requires reading something they cannot quickly act on. There is nothing to approve. There is nothing to forward. There is only a question to take back to the document.

The third part is adjudicating exceptions. The override log is the highest-signal artifact in the building, and it is the leader's job to read it. Every override is a moment where a human disagreed with the system. Some are noise. Some are pattern. Some are evidence that the intent document needs a new section. The leader's job is to sort them.

The override log is also where junior people contribute upward. The junior attorney on Priya's team who caught a data residency issue on her way out the door on a Friday now logs the override with her reasoning. Priya reads the log. The override surfaces a class of cases the charter did not cover. Priya updates the charter. The junior person's catch has become a structural change. That feedback loop did not exist in the old model. In the old model, the catch was a heroic individual save that got told in hallway conversations and forgotten.

The fourth part is hiring for judgment at the edge, not execution in the middle. The middle is what the system does now. Drafting. Summarizing. First-pass analysis. Filtering against criteria. The leader is no longer hiring people primarily to do that work. She is hiring people primarily to operate at the edge of what the document covers: the cases the document did not anticipate, the patterns the system surfaces that nobody named yet, the trade-offs that emerged from a customer conversation last week and have not been written down.

This changes the interview. The standard interview for a senior analyst, three years ago, tested whether the candidate could produce a clean piece of work product within a defined scope. That work product is now produced by the system. The interview that matters tests whether the candidate can spot when the document is missing a section, can write a clear sentence about a hard trade-off, and can disagree with the system in a way that contributes to the next version of the document. None of that is easy to screen for, and most of the standard interview tools are oriented toward the old job.

Priya's team rewrote their senior attorney job description after the intent architecture exercise. The previous version was a generic list of legal competencies. The new version described a specific kind of thinking: working at the boundary of stated risk positions, recognizing when those positions are stale, articulating a new position clearly enough that the system can be reconfigured against it. She told me hiring against the new description was harder. The pool was smaller. The candidates who came through it stayed longer.

The Pushback I Owe the Reader

The obvious read of "fewer translators, more senior judges" is headcount reduction. Some readers will mentally translate the chapter into a 20% cut and a flattering Q3 earnings call. That reading deserves a direct response.

A friend at one organization described his company treating the intent architecture work as an efficiency project. They encoded constraints. They deployed the system. They reduced headcount by about twenty percent, mostly in the layer that had been doing translation. They moved on. Within four months, output quality had degraded noticeably. They blamed the AI vendor. The actual failure was that nobody had the seniority or the bandwidth to notice when the constraints went stale. The documents were running on autopilot. The villain came in through the door the company had left open.

The intent document is not self-maintaining. It needs an owner with time, a leader reading the drift report, an override log being adjudicated. None of that work is visible in a productivity metric. It produces conditions for the system's output to remain correct. If you cut the people doing it, you do not get a leaner organization. You get a system efficiently producing yesterday's company at a scale that hides its drift until something breaks.

The honest version is fewer people doing translation, more people doing the judgment the document cannot pre-answer. Not fewer people overall. The math depends on the function and on whether the former translators can move into the senior roles, which some can and some cannot. If the new operating model is sold internally as an efficiency play, it will be implemented as one. If it is sold as a clarity play, with a real investment in the people who maintain the documents, it has a chance.

A Calendar That Is Quieter and Harder

The CRO in Chicago, six months in, told me his calendar had fewer meetings on it but more conversations. His weekly pipeline review was now forty-five minutes. He had added a Monday-morning drift review, a Thursday override log read with two of his directors, and a monthly conversation with the head of marketing and the head of customer success about whether the qualification criteria still matched the company's commercial strategy. Those last conversations were the ones that had no defined endpoint. They sometimes ran two hours. They sometimes ended unresolved.

His total meeting hours were down. His effort was up. He told me, more than once, that the new shape of the week was less satisfying in the moment and more satisfying in the aggregate. He could see, looking back, that the company was making better decisions than it had been six months earlier. He could not always tell, looking at any given Wednesday, whether the day had gone well.

That is the texture of the new job. Less legible to the leader who is living it. More legible to the organization a year later, when the meetings have gotten shorter and the decisions have gotten cleaner and the new hire ramps in two weeks instead of two quarters.

The conceptual move is one thing. The implementation, in a company that has eighty-three deployed AI workflows and a strategy off-site coming up in March, is another. The next chapter is the place. The 90-day playbook is not a transformation framework. It is a way to do the smallest version of this work first, find out where it breaks, and then decide whether to do more.

Tool: The Override Log

Use the override log whenever a human corrects what an AI workflow produced.

The log is a maintained record of every override. Fields:

- Date
- System (which workflow or agent)
- Intended output (what the system produced)
- Actual override (what the human changed it to)

- Reason given (one or two sentences from the human who overrode)
- Status (one of: pattern to learn from, one-off, intent doc update needed)

Example row:

Date	System	Intended output	Actual override	Reason given	Status
2026-03-04	Pipeline qualifier v3.2	Disqualified deal #41128 (mid-market healthcare, no signed BAA) for failing data residency criterion	Re-qualified, moved to "human review" tier	"Deal includes a regional hospital that meets the spirit of the residency policy but not the letter. The policy as written does not contemplate this hospital structure."	Intent doc update needed (residency policy needs a section on regional hospital exceptions)

Read the log weekly. Recategorize anything marked "one-off," and update the intent document on the rows that point at structural gaps.

12

CHAPTER 12

The 90-Day Transition

A practical playbook, and what to do when it goes sideways

"Our Monday two-hour operations meeting has become unnecessary, and I do not know what that means."

That is what a supply-chain COO at a mid-sized consumer goods company told me on a Tuesday in February 2026. The meeting had been the spine of her week for four years. Twelve people. Status updates from every region. Escalations. Exception handling.

Three months earlier, her team had stood up an intent-driven exception-handling workflow against a charter she had drafted with her ops leads. By Sunday night the system had triaged the week's exceptions against the charter, surfaced the ones requiring human judgment, and produced a summary the leads could read in twenty minutes.

The meeting had nothing left to do. The escalations had already been adjudicated. The status updates were redundant. She kept holding it for three more weeks out of inertia. Then she canceled it.

That is what successful 90-day transitions look like at the end. Not a transformation deck. A meeting that no longer makes sense.

Most do not end that cleanly. The playbook is more useful when you know where it breaks.

The 90 days break into three phases. Each is thirty days. Each has one job. None of them are quick wins.

First 30 Days: Find the Ceiling

Diagnose. Do not deploy.

Build a workflow inventory. A real spreadsheet. Every place in your function where someone is prompting an AI system as part of regular work. Person's name. Frequency. Output. Rough time per cycle. This inventory exists nowhere in most companies. I have heard about seven leadership teams running it. Every one found at least three workflows leadership did not know existed. One CRO told me he had discovered his sales operations analyst had been running a weekly prompt chain to produce his pipeline report for six months. The CRO had been reading the report every Monday and assuming it came out of the CRM.

Rank by volume, not by complexity. A simple prompt run 200 times a week beats a complex prompt run twice a month. The simple high-volume prompt carries the heaviest translation tax.

Find the repeated translation. Read the prompts side by side. When three people open with the same five sentences about what the company wants, those five sentences are an intent document nobody has written down. That is your candidate.

Name your Derek. Not by title. By dependency. Ask the team: if X went on vacation for a month, what would break. The first name they say is your Derek.

Select one pilot. One function. One workflow. Not three. Not the most ambitious one. The one with a real Derek, a clear translation pattern, a willing senior owner, and enough volume that the maintenance loop will run inside 90 days.

The mistake leaders make in the first 30 days is doing the inventory and skipping straight to action. The inventory is the action. A sloppy inventory picks the wrong pilot, and the wrong pilot produces results that confuse you for months.

Checklist for the first 30 days:

- Workflow inventory complete, with named owners and frequencies
- Three highest-volume prompt-based processes identified

- Translation patterns documented (which prompts repeat which framing)
- Derek identified by name
- Pilot function and workflow selected with a senior intent owner committed to the next 60 days

The supply-chain COO told me later that the inventory alone had already paid for itself. She had found a workflow her finance lead had been running quietly that touched data she had assumed was handled by a different team. The inventory was the first time anyone had asked the question across functions. Nothing in the rest of the work had been as valuable as that one Tuesday afternoon spent listing everything.

Days 31 to 60: Write the Intent

Month two is the document. The hard month. Most projects fail here, not because the writing is hard, but because the disagreements the writing surfaces are.

Run an intent extraction workshop. Not a strategy off-site. Not a vendor presentation. A working session with the senior people whose judgment the system is supposed to represent. Full day. Three to six people. Bring food. Take laptops away from anyone who is not the scribe.

The workshop will go badly the first time. Plan for it. Senior people in a room trying to write down what their function actually wants will produce sentences that say nothing, get stuck on a question that turns out to be load-bearing, and end the day exhausted. That is the workshop working correctly. The asset is the surface of the disagreements, not the finished document.

Capture real trade-offs, not generic ones. A generic trade-off is "we value speed and quality." A real trade-off is "when speed and quality conflict, we will accept up to a 4% increase in revision rate to keep our 24-hour turnaround for our top 20 customers, and we will not extend that turnaround for customers outside the top 20 even when they ask." Write down the things you are willing to lose.

List the open decisions. The places where senior people actually disagree and have been avoiding the disagreement. You do not have to resolve all of them in the workshop. You do have to write them down as open questions, with a named owner and a date by which they will be answered. The open-questions section of an intent document is often more useful than the answered section, because it tells you what the company has not yet decided.

Define revisit conditions. Every section gets a trigger for review. A date, an event, or a metric. The data residency section revisits quarterly or when EU regulation changes. The brand voice section revisits when share of voice in a competitor set moves more than a defined threshold. The qualification criteria revisits when win rate in the top three verticals moves more than a defined amount. Without revisit conditions, the document goes stale within a quarter and the villain comes back through the side door.

Assign ownership by name. Not a team. A person. The person is responsible for answering "is this still what we believe" on the revisit cadence. If the person leaves the company, the section is unowned, which is its own escalation.

Connect the document to one workflow. The pilot. The document gets loaded into the system's context at the execution hook. The technical work is usually trivial. The political work, getting the senior owner to agree that the document is final enough to be machine-readable, is not. Most senior people want to add "for now" to every sentence. Resist. The point is a snapshot the system can run against. The snapshot will be wrong in places. You will fix it in month three.

Teams that skip the open-questions section, the revisit conditions, or the named owner discover three months later that the document has papered over a disagreement nobody surfaced in the room. The redo costs the better part of a quarter.

Checklist for days 31 to 60:

- Workshop held, with the right people in the room and laptops away
- Trade-offs captured in specific language (specific numbers, specific cases)
- Open questions section drafted, with named owners and dates
- Revisit conditions defined for every section
- Section ownership assigned by name
- Document loaded into the pilot workflow's context through an execution hook

---

Days 61 to 90: Operationalize and Maintain

Month three is where most playbooks stop and the actual work begins. The 90 days do not finish the transition. They finish the first cycle of a loop you will run forever.

Run the pilot on real work. Real volume. Not a sandbox. If you cannot run the pilot on real work, you have picked the wrong pilot and you are five weeks behind. Go back.

Log every override. Every time a human corrects the system's output, the correction goes in the override log. Date. System. Intended output. Actual override. Reason. Status: pattern to learn from, one-off, or intent doc update needed.

The override log will look chaotic in the first two weeks. The team will mark almost everything "one-off." Have the intent owner read the log at the end of week two and recategorize. About a third of the entries will move from "one-off" to "pattern" or "intent doc update needed." That is normal.

Review drift weekly. The drift report shows where the system's aggregate behavior is moving away from the intent document. Not error rates. Drift. Put the review on the intent owner's calendar before the pilot starts. If she misses a week, the cycle has already started to break.

Update the document. Every two weeks, the intent owner takes the override entries marked "intent doc update needed" and edits the document. Each edit gets a date, a changelog entry, and a one-sentence reason. The system reloads on its next run.

Measure decisions changed, not just time saved. The most common reporting mistake at day 90 is showing leadership how many hours the automation saved. Those hours are a subset of the return. What you want to measure is whether the decisions the function makes have changed. Are different deals being qualified. Different contracts pursued. Different content prioritized. If only the speed has changed, you have moved the prompt-and-response ceiling to the workflow layer. If the decisions have changed, the function has moved.

Priya measured decisions. Four months after her analytical charter was operational, she could point at three contract structures her firm had pursued that previous-her would have walked away from, because the charter had made explicit a risk-acceptance threshold the firm had been treating informally. Those three contracts were the return. The 90 minutes of meeting time saved was the appetizer.

The supply-chain COO measured something simpler: the count of weeks in a row the Monday meeting was canceled. After eight straight weeks she stopped putting it on the calendar. She told me she felt a small, real sense of loss. The meeting had been hers.

The 90 days do not feel triumphant at the end. They feel quiet. If at day 90 you are reading override logs on a Tuesday afternoon and updating a document section nobody else noticed needed updating, you are doing it correctly. The visible signs of progress will lag by another quarter or two. The internal signs

(the meeting that stopped mattering, the analyst who flagged a pattern she could not have flagged before, the conversation where you both reached for the document instead of relitigating from memory) will start showing up by week ten.

Checklist for days 61 to 90:

- Pilot workflow running against the intent document on real work
- Override log live, with the right status categories, recategorized at week two
- Weekly drift review on the owner's calendar
- Biweekly document update cadence established, with changelog
- Decisions-changed metric defined, and at least one baseline data point captured

The Failure Modes Worth Naming

Three failure modes show up more than once.

The maintenance skip. Team writes the document in month two, deploys it in month three, declares victory, stops touching the document. Within four months the document is stale, the system is running on yesterday's assumptions, and the function is back where it started with the addition of false confidence. The loop is the work.

The headcount cut. Leadership reads the playbook as an efficiency story and treats month three's deployment as the moment to reduce the translation layer. The pattern is consistent: short-term margin improvement, four to six months of degrading output quality, and an eventual realization that the cut layer was holding the document together. Recovery takes longer than the original transition.

The wrong pilot. The team picks a glamorous workflow that gets attention from leadership but does not have enough volume to exercise the maintenance loop, or lacks a real Derek whose translation work is being externalized. The pilot looks fine. It produces a document. The document is never tested by real exception flow. When the team tries to extend the model to a high-volume workflow six months later, nothing in the pilot has prepared them for what maintenance looks like at scale. They start over. Volume matters. Disagreement matters. The pilot has to hurt enough that the loop has to run.

The Question the Book Has Been Building Toward

You arrived at this book with a feeling. Something about your AI deployment was not wrong, exactly, but was not enough. You had done the work. Your team had done the work. The pilots ran. The CFO signed off. People were using the tools. And somewhere in the back of your operational instincts, something was telling you the return did not match the investment. Not the financial return. The organizational return.

I said in the first chapter that the dissatisfaction was accurate. A signal, not a personal failure. It was also incomplete. It pointed at the right problem and framed it as a technology problem. The organizations whose leaders described doing this work well arrived at a different question.

The question is not "how do we get more out of our AI tools."

The question is: have you ever said what you want clearly enough for any system to act on it?

That word "system" is doing real work. Not just AI systems. Any system. New hires. Partner organizations. Outsourced teams. Internal processes that run on institutional memory and the presence of two or three people who know how things really work. The AI made the problem visible because the AI will not fill in gaps the way a person will. It does not guess right based on context. It does not absorb your culture through a year of sitting in meetings. It takes what you give it and produces output at a scale that makes every unstated assumption show up in the work, at volume.

The problem predates the AI.

Sana said it better than I could. "The AI didn't make our voice worse. It made us realize we'd never finished defining it. I'm not sure we ever will." That last sentence is the one I keep coming back to. I'm not sure we ever will. The intent-driven organization is not a project with a completion date. Sana started with eight constraints and added two more three months later. Her document is still being written. It will always be still being written.

The prompt era was necessary. It taught organizations how to talk to these systems at all, how to evaluate output, how to absorb a new tool into existing work. None of that is wasted. The companies that skipped the prompt era did not arrive at the next operating model faster. They arrived at it confused.

The prompt era is not enough anymore. A faster version of yesterday's company is still yesterday's company. The teams that have lived inside the prompt-and-response model for two years know this in their bones, even when they cannot say it. The dissatisfaction is the knowing.

The next operating model is intent-driven. Strategy versioned, not just presented. Trade-offs written down, with named owners and revisit conditions. Documents the systems read before acting. Drift reviewed weekly. Overrides treated as signal. Judgment hired for at the edge.

Leaders who keep intent in their heads become bottlenecks. Every decision routes through them. Every ambiguity sits on their calendar. Every system runs on what they happened to say in the last all-hands and what they have not said yet. The organization moves at the speed of their attention, which is a constraint that scales with neither headcount nor compute. Leaders who externalize intent get something the others do not: an organization that learns faster, drifts less, and survives their absence.

The first step is small. Pick one workflow. Write the intent. Connect it through an execution hook. Read the override log. Maintain the document. You will not transform your company in 90 days. You will run the first cycle of a loop you will run for as long as the company exists.

You have a blank document. You have ninety days. The document you produce will not be good. It will be incomplete, contradictory in places, full of questions you cannot answer alone. It will be more useful than anything your AI vendor has ever sold you, because it will be a precise record of what you know, what you don't know, and what you have been avoiding knowing.

The document is the work. Start it.

The next advantage will not belong to the companies with the best prompts. It will belong to the companies whose systems know what the organization actually wants.