WHITEPAPER · AGENTIC AI FOR PRODUCT LEADERS

Designing for the Supervising User

The half of an agentic product that ships to the person watching

Every rung up the autonomy ladder converts more of your users into supervisors of a job they used to do themselves. They are not a role you staff. They arrive on launch day, and their competence is a property of what you shipped.

Yoram Friedman, MD
Physician and enterprise product leader · SAP, Walmart, Microsoft, regulated health technology
Drawn from the Agentic AI for Product Leaders series · agenticaiproductmanagement.com

THE ARGUMENT

Somebody is watching your agent. You have probably never met them.

An agentic product ships two things. The agent, and the experience of supervising the agent. The first has a roadmap, an owner and a demo. The second is usually assumed to be somebody's problem after launch.

It is not, and the reason is simple enough to state in one sentence. The second product has a user.

When a product climbs the autonomy ladder, the person who used to do the work does not disappear. They are converted into the person who watches the work being done. That is a different job, with a different skill, and nobody trained them for it because nobody noticed it had happened.

> You do not hire this person. You do not have a headcount ratio for them. They arrive on the day you ship, in numbers, because they are your users.

This document is about that person: who they are, why supervision fails in two entirely different ways, what you actually have to build for them, and how to tell in production whether any of it is working.

PART ONE

The role migration nobody planned

Autonomy is not a yes-or-no question. It is a ladder, and every product sits on a rung. Low down, the system suggests and the person decides. Higher up, the system acts inside a boundary and the person reviews. Higher still, the person sees only the exceptions, and eventually only the report.

Every rung moves authority to the agent. That much is well understood, and it is where most of the design attention goes. What gets far less attention is that the same movement quietly rewrites the human's job.

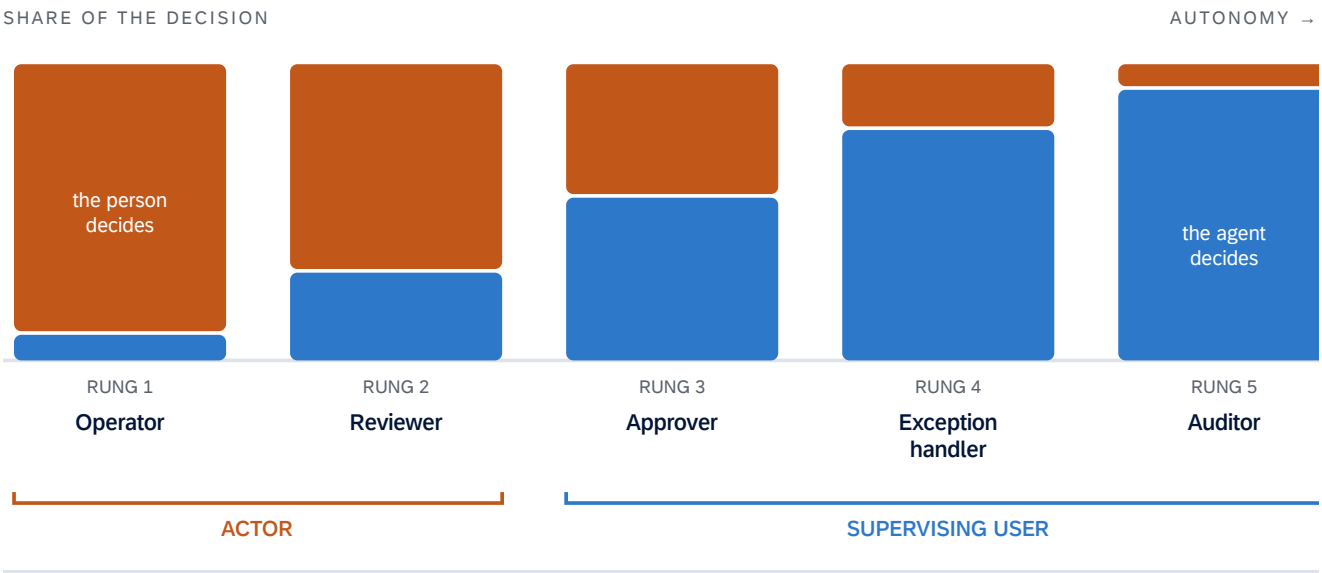

Figure 1. The same ladder, read from the other side. As the agent's share of the decision grows, the person's job does not shrink; it changes kind. Somewhere around the middle rungs they stop being an actor and become a supervisor, usually without anyone announcing it. Schematic, not measured.

The shift is cognitive rather than procedural, and it cannot be trained in a week. An actor focuses on the single case in front of them, executes the task, reviews that case, and relies on domain judgment. A supervisor watches patterns across many cases, approves and intervenes and tunes, catches drift before it compounds, and relies on judgment about judgment. Those are different skills and only one of them was in the job description.

The failure mode is predictable and it has two shapes. People keep doing the old job and abandon the supervisory one, or they slide into passive monitoring that catches nothing until it has already compounded. Both look identical on an adoption dashboard, which is why the dashboard will report that everything is fine while the supervision quietly fails.

Why this is not a staffing question

A second supervisor is emerging inside companies, usually called AgentOps. It watches the fleet: drift, hallucination rate, latency, cost per task, whether the thing that passed evaluation in March is still correct in November. It is the agentic equivalent of what site reliability engineering became for cloud software, it is real, and on most teams it is unstaffed.

The supervising user is not that. They sit outside your building. There is no ratio to argue about, no requisition to open, and no way to decline to have them. The only decision available to you is whether they arrive into a product that was built for them.

PART TWO

Two kinds of oversight, and the case where one did nothing

"Human in the loop" names two different commitments and hides them inside one phrase. Assuming the first covers the second is the error that regulators and courts are now treating as no oversight at all.

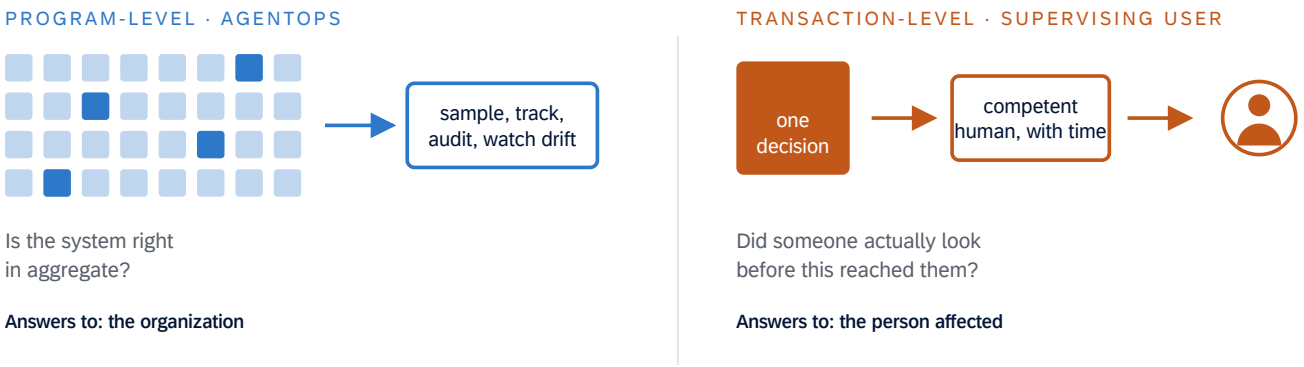

Figure 2. Program-level oversight asks whether the system is right across thousands of decisions. Transaction-level review asks whether anyone looked at this one. Different questions, different owners, different failure modes. You can satisfy the left completely and have nothing of the right.

Program-level oversight is review of the system in aggregate. A team samples outputs, tracks success and failure rates, watches for drift, and audits performance against the standard the product committed to. This is AgentOps work, and it produces a number an organization can defend.

Transaction-level review asks something narrower and harder. Before this specific output changes this specific person's life, did a human with the competence and the time actually look? This is the supervising user's work, and it produces nothing an organization can point to, which is exactly why it is the half that gets quietly removed.

Three hundred thousand claims, ninety days

A health insurer ran an AI system to review medical-necessity claims. A physician signed each denial, so on paper there was a human in the loop on every single decision. Internal records surfaced in litigation later showed physicians signing off on more than three hundred thousand claims in a two-month window. The same litigation alleged that of the small fraction of denials patients appealed, around nine in ten were overturned.

1.5 sec

The time available per claim at that volume. Program-level oversight existed and could produce a defensible aggregate number. There was a human in the loop on every one of those denials and no human oversight of any of them.

A second and a half is not a review. It is not even a glance. It is a signature, and the system counted it as oversight. Nobody designed that outcome. It is what happens when a product ships to a population that has been converted into supervisors and nothing in the product was built for the conversion.

The design question this forces is concrete, and it is per output class rather than per product. For each thing your agent produces, which of the two kinds of oversight does it require, and what stops one from masquerading as the other?

Regulation has already taken a position. The EU AI Act's Article 14 obligations are transaction-shaped rather than aggregate-shaped, and Article 14(5) goes further for one narrow class, remote biometric identification, by requiring that the action be verified by two separate people. California SB 1120, signed in 2024 and in force since January 2025, requires that medical necessity be determined by a physician rather than by an algorithm. In both cases the law is describing the supervising user's job and, by implication, requiring that your product make it possible to do.

PART THREE

Supervision fails on two clocks

A fix aimed at the wrong clock does nothing. One failure is present the day you launch. The other takes eighteen months and starts out working.

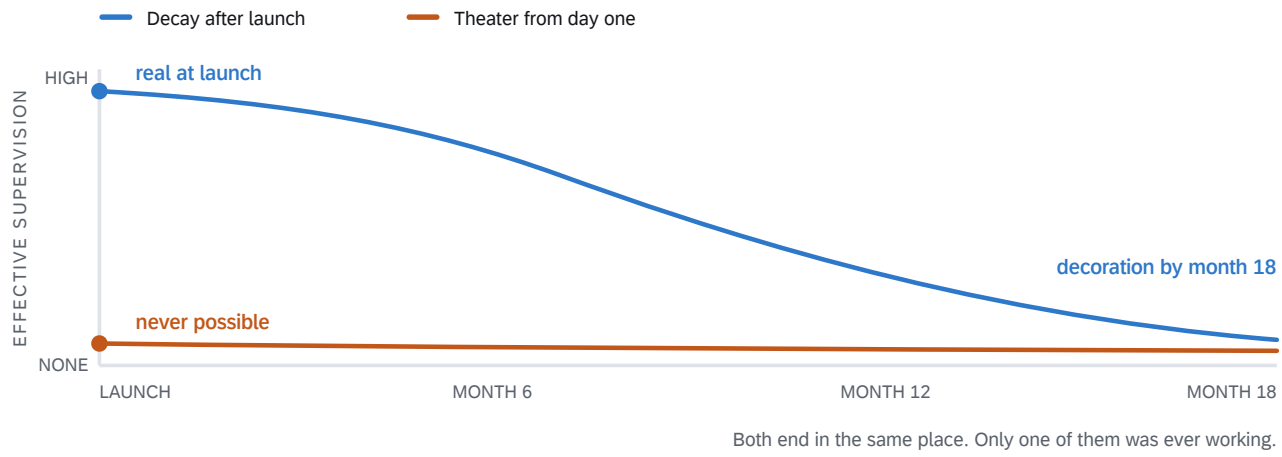

Figure 3. The two failure curves. Theater was never supervision: the volume, the interface or the incentives made review impossible from the first day. Decay was real and stopped being real. Only one of them can be fixed by anything you do after launch. Schematic, not measured.

Theater from the first day

The first failure ships with the product. Review was never possible, because the volume, the interface or the incentives made it arithmetically impossible. The insurance case is this failure. So is any queue where the throughput expected of the reviewer, divided by the working day, yields a number of seconds per case that no competent person could use.

This is not a behavior problem and no amount of training will fix it. It is a capacity problem you designed in, and the useful thing about it is that it is measurable before you ship. Expected volume divided by supervisor hours gives you seconds per decision. Look at that number without flinching and you will know, before launch, whether you have built supervision or its appearance.

Decay after it

The second failure is harder, because the system starts out working. Three mechanisms compound, each with its own literature, all firing at once.

Warning fatigue arrives at the interface. Studies of interrupting security warnings found dismissal rates around seventy percent, and an agent's output does not even interrupt; it sits inline in the flow of work, so the effective dismissal rate is higher than for the warning that at least made someone click.

Automation complacency arrives at the workflow layer. Vigilance falls as trust rises, and it falls fastest on the systems that are most reliable. Lisanne Bainbridge named this in 1983 as the irony of automation: the better the automation, the more thoroughly atrophied the operator's skill at catching it when it fails, and the rarer and harder the cases where they must.

Automation bias arrives at the decision layer. The supervisor stops treating the agent's output as a proposal and starts treating it as the answer, which means the review is confirming rather than checking.

> Deference fails in both directions. A supervisor who accepts everything is not supervising. A supervisor who overrides everything has stopped using the product. If nothing in your design helps them decide which to do this time, they are deferring by habit, and habit is the one thing deference cannot afford.

There is a fourth failure that hides the other three. Users who learn the agent's quirks build shadow workflows around them: the manual double-check, the exported spreadsheet, the message to a colleague who knows. Task metrics stay green because the work still gets done, while the supervisory system you shipped is being routed around. Compensatory behavior is the most under-instrumented signal in agentic products, and it is the one that tells you the design failed while everything still looks fine.

PART FOUR

What you actually build

Channel 1 is the agent: its autonomy boundary, logging, error handling and recovery. Channel 2 is the human experience of supervising it. Channel 2 is a product, with requirements, a design and a budget, and it has four dimensions that most teams treat as one.

ONE · TECHNICAL

Can a human see, pause, override and undo what the agent is doing, in time? Typed interruption with durable run state and an explicit approve-or-reject resume. A kill path that lives outside the agent's own runtime, so a misbehaving agent cannot suppress it. Rollback with a time-to-restore that has actually been measured in production conditions rather than in a drill.

TWO · ORGANIZATIONAL

Who is doing the supervising, and is the volume compatible with the attention it needs? This is where the interruption budget belongs: a cap on the number and priority of approval requests a supervisor receives per unit time, with routing, deferral and batching against that cap. Without one, the number of approvals is set by however many the agent happens to generate, which is a number nobody chose and which grows with adoption.

THREE · REGULATORY

Can the decision be reconstructed and defended six months from now? A decision record carrying inputs, reasoning, output, confidence, timestamp and a pinned model version. An appeal path that someone outside your team can actually use. If the answer to "why did it do that" is "the agent did it," the audit surface failed before the question was asked.

FOUR · MORAL

Who answers to the person the decision landed on? That person is often not your user, has no account, and appears in none of your analytics. Name the owner of affected-person outcomes in advance, not on the night it breaks.

A team that has built only the first dimension has built roughly a quarter of Channel 2 and usually believes it is finished.

The approval moment is a design artifact, not a dialog

Where the agent must stop and ask, what it hands over when it does, and what the human is actually being asked to decide. A confirmation box that says "proceed?" transfers liability without transferring understanding. The unit that works is a decision package: what the agent concluded, what it relied on, what it is uncertain about, and what happens if the answer is no. Every major agent framework has converged on the same primitive for this, a typed interruption object with durable state and an explicit resume, so the mechanism is available. What is missing is the product decision about what goes inside it.

Make disagreement as cheap as agreement

If accepting the agent's proposal is one click and overriding it requires a free-text justification, you have priced deference into the interface and you will get it. Symmetry here is a design choice with a measurable downstream effect, which brings us to the last question.

PART FIVE

Telling whether any of it is working

No platform ships supervisory metrics as named primitives. The trace data exists almost everywhere and the metrics are derivable from it, but the composition is your team's job. An instrument you cannot produce at all is not a gap in your data pipeline. It is a finding about a surface nobody designed.

Six instruments describe the agent. Read from the supervisor's side, the most useful of them is **override frequency**, and the thing to watch is the band rather than the level. Drifting low reads as the agent improving, and it is equally consistent with the supervisor having stopped looking. Drifting high means they have stopped using the product. Both directions are bad news and they mean opposite things.

The others each carry a trap. Task success is not completion, and completion is the one that is easy to collect, so completion is the one that gets reported. Unintended action rate requires validating the state of the world, because semantic validation will happily pass an output that says "done" when nothing happened. Confidence calibration matters because an uncalibrated score is worse than none, since it directs attention away from the cases that needed it. Rollback time measured in a drill is a number about your test environment. And incident recovery time usually excludes reconstruction, which for multi-agent systems is most of it.

Four more sit one layer down and describe the supervisor rather than the agent. Time to intervene, from the agent surfacing something to a human acting on it. Response rate per interruption priority, which tells you whether your priority tiers mean anything or whether everything is treated the same. Review depth, the time actually spent per decision. And the split between work going through the supervised path and work going around it, which is how you detect the shadow workflows. None of these ships as a named metric anywhere. All are derivable from trace and span data plus whatever your human workflow system records.

If you build only one, build this one

Review depth: time actually spent per supervised decision, distributed rather than averaged, with the fastest decile called out.

The average will look reasonable in almost every failing system. The bottom decile is where the signature-not-a-review failure lives, and it is visible from the first week of production rather than after the incident. It is also the one number that would have caught the insurance case on day one.

THE ONE LINE

The supervisor is not a role you staff.

It is your user, one rung up the autonomy ladder, doing a job nobody gave them. Their competence is not a training problem you can hand to operations after launch. It is a property of the product you shipped, and it is the half of the product that decides whether the other half is safe.

Sources and status. Drawn from the *Agentic AI for Product Leaders* series by Yoram Friedman: *Why Agentic AI Products Fail*, chapter 8 on the two kinds of human-in-the-loop, chapter 15 on deference, chapter 16 on the actor-to-supervisor transition, chapter 17 on why human-in-the-loop fails, and chapter 18 on skill erosion; *Agentic AI for Busy Product Managers* (second edition) on the autonomy ladder, the four runtime artifacts, the six instruments, the interruption budget, and the supervisory engagement metrics including time-to-intervene and review depth; *The Agentic AI Team*, on who owns the operation of Channel 2; and *The Agentic AI Practitioner* (forthcoming, September 2026). The medical-necessity claims case is drawn from public litigation records, described by role rather than named, and the figures within it are allegations rather than findings. The irony of automation is Lisanne Bainbridge, 1983. The warning-dismissal rate is from the security-warning literature and is directional for agent interfaces rather than measured on them. Regulatory references are EU AI Act Article 14 and California SB 1120. Figures 1 and 3 are schematic illustrations of the argument, not plots of measured data. Terminology note: this paper uses *AgentOps* for the internal function that watches the fleet and *supervising user* for the person the product converts into a supervisor; the published books use "agent supervisor" for the former.