The Support Machine

How to Deploy AI in Customer Support Without Breaking Trust

Barnir

Reader's Note

This book began as a live field journal from inside a real AI support migration. The point was not to write a polished case study after the fact. The point was to capture the work while it was still alive, when the risks were unresolved, the machine was not yet trusted, and every confident sentence could still become a cautionary tale.

The central argument is simple: AI support is not a chatbot project. It is an operating model migration.

A chatbot answers. A support machine routes, reasons, retrieves, escalates, measures, learns, and changes the work humans do around it. That makes the launch more powerful, and more dangerous, than most teams admit.

The metaphors in this book are intentional. The Rubicon, the machine, the goalkeeper, the wizard, the tollbooth, the shadow run, and zero hour are not decorations. They are operating language for decisions teams usually bury under vendor decks and roadmap slides.

Who This Book Is For

This book is for support leaders, CX operators, support engineers, customer success leaders, technical founders, and executives who know the AI demo is easy and the launch is hard.

It is especially for B2B SaaS teams whose customers are technical enough to notice when the machine is bluffing.

It is not for people looking for a generic tour of AI trends. It is not a model-training manual. It is a field guide for operators who have to protect trust while changing the physics of the queue.

How To Use This Book

Read it once as a story. Then read it again as a migration plan.

The story matters because AI support is full of false confidence. If you only read checklists, the work looks cleaner than it is. If you only read the narrative, the work feels more mysterious than it is. The useful truth sits between the two: operators need a narrative strong enough to align the company and artifacts sharp enough to survive launch day.

Each part of the book moves through a different layer of the migration. Part I diagnoses the friction. Part II walks through the market and the minefield. Part III designs the machine. Part IV launches it. Part V turns launch into an operating model.

The field notes are deliberately blunt. They are the lines you should be able to repeat in a planning meeting, vendor review, launch room, or executive update. Each one turns the chapter’s story into an operating lesson to carry forward..

The Operating Map

The book uses six gates. A team that cannot pass a gate should not pretend it is ready for the next one.

1. Gate 1: Diagnose the queue. Know what customers ask, what they cannot find, and which questions are truly solvable.
2. Gate 2: Choose the machine. Separate real systems from demos, wrappers, hidden humans, and vendor hand-waving.
3. Gate 3: Design the guardrails. Decide what the machine can know, say, do, and stop doing.
4. Gate 4: Shadow-run the machine. Test it against real cases before customers can feel the blast radius.
5. Gate 5: Launch with direct accountability. Every lane needs a named owner. No orphaned decisions.
6. Gate 6: Operate the living system. Review failures, ship fixes, update knowledge, and promote humans into system work.

Contents

Introduction: This Is Not a Chatbot Project

There is a dangerous sentence hiding inside almost every AI support project: "Let's add a bot." It sounds small. It sounds practical. It sounds like a feature. It is usually the first sign that the team is underestimating the migration.

A bot is a surface. A support machine is an operating model. The difference matters because customers do not experience your org chart. They experience the answer, the escalation, the handoff, the policy, the missing document, the broken workflow, the tone, the timing, and the confidence of the system when it says it knows what to do.

If the machine is wrong, trust burns faster than time is saved. If the machine is right, the support team stops being a queue-clearing function and becomes the architecture team for customer knowledge.

That is the real promise of AI support: not fewer humans, but better-shaped human work. The machine should absorb repetition, expose product gaps, route ambiguity, and make the organization smarter every time a customer gets stuck.

This book follows one live migration from friction to zero hour. It keeps the mess in the frame because the mess is the point. The vendor demo is not the hard part. The hard part is deciding what the machine is allowed to know, what it is allowed to do, when it must stop, who owns the failure, and how the humans around it get promoted instead of erased.

The book is closed, but the story is not. You pay for it once (it's free) and then, if the machine keeps learning, the text should keep learning too. The next version ~~can~~ will become a living skill: a book that updates as the operating model changes and eventually helps readers run the migration instead of only reading about it.

The Operating Thesis

- AI support is not a chatbot project. It is an operating model migration.
- The goal is not deflection. The goal is a trustworthy resolution.
- The support engineer does not disappear. The support engineer becomes a system architect.
- The queue is not only a cost center. It is a signal stream for product, documentation, onboarding, sales, and customer success.
- Launch day is not the end. Launch day is when the machine starts producing evidence.

PART I

THE FRICTION

Why good support metrics can still hide a scaling crisis.

Every AI support story starts before AI. It starts with a queue that looks manageable from far away and expensive up close.

The trap is that healthy support metrics can become a sedative. CSAT looks good. SLA looks good. The first response metric looks good. The team looks responsible. Then someone asks the better question: why are brilliant support engineers still repeating answers the product, docs, and onboarding should have made unnecessary?

This part is about that moment. The mirror is uncomfortable because it shows the difference between performing well inside the old system and needing a new system entirely.

1. The Friction Point

The mirror is a cruel invention.

Sentry exists for one reason: to help developers see what is broken in their code. We are the “truth” platform. The check engine light. The thing you look at when things go sideways.

But when we turn that lens on our own support workflow, the reflection isn’t pretty.

We have a friction problem.

The “Good Enough” Trap

Here’s the thing about metrics: they can lie by omission. Our CSAT scores are over 95%. Our SLA is touching the 98% mark. First time to respond, number of touches? Top performing. If you look at a dashboard in a QBR, you nod and move on to the next slide.

But doing things “as expected” is death in Silicon Valley.

Beneath the “as expected” metrics, our support team is re-educating customers and prospects on what our documents and onboarding experience already cover well. Our product velocity, the sheer speed at which Product and Engineering ship new features, is outpacing our ability to support them.

We are throwing bodies at a physics problem.

The Breaking Point

We realize that adding headcount isn’t scaling linearly with ticket volume, or with the increasing quality, sophistication, and urgency our customers require. We are asking brilliant support engineers to answer the same “What is a DSN?” question, instead of thinking about how our developer customers want to find answers on their own (which is already here: https://docs.sentry.io/concepts/key-terms/dsn-explainer).

That is not Support Engineering; it’s expensive copy-pasting.

“What is Sentry?” is something I discuss at dinner parties (I’m a fun guest), but I can’t stop asking myself: why aren’t developers finding the answers already sitting on this page (https://docs.sentry.io/product/) on their own?

We have to admit a hard truth: our technical velocity is elite, but much of our tooling are stuck in 2015. We can’t stay still because eventually the wave of interactions with customers volume is going to crash over our heads.

So we decided to stop buying umbrellas and start changing the weather.

Field note: *If your support metrics look healthy but your team is re-answering the same questions, the dashboard is hiding the actual problem.*

2. Do We Actually Need AI, or Just Better Operations?

I have played every character in AI tragedy and successes

I haven't just watched this wave from the beach. I've drowned in it.

I have been the **Builder**, writing the specs. I have been the **Implementer**, trying to glue the API to a legacy database. I have been the **Ambassador**, selling the dream on stage. I have been the **Owner**, signing the checks. I have been the **Consultant**, billing by the hour to explain AI to various audiences.

Across every single one of those roles, from startup chaos to enterprise meeting rooms, I learned one universal, painful truth:

AI is incredible. Outcomes are not.

There is a specific kind of noise you hear in AI companies. It's the sound of people conflating "potential" with "production." It's the quiet hum that tells you something is "working" on a demo screen while the actual operational reality is slowly misaligning.

For a long time, I treated that hum as normal. I thought confusion was just the tax you paid for innovation. I was wrong, and documented the philosophy behind it, here: https://barnir.substack.com/s/the-ai-upgrade.

Those moments weren't "friction." They were warning signs. Looking back, the patterns are identical whether you are selling the tool or buying it.

Here is what I ignored then, and what we are refusing to ignore at Sentry now.

1. The Illusion of Motion The first warning sign is the easiest to ignore because it feels productive: the belief that *shipping in the world of Support* means *working*.

AI teams are addicted to this. Iteration speed is a survival strategy; velocity creates a sense of safety. When something goes out the door, the internal slack channel claps. It feels like progress.

But output is a dangerous metric.

The early signal is that you keep saying "yes" to launches, yet customers don't change their behavior. Internally, the product is evolving. Externally, nothing moves. Nobody cares.

That gap is where AI initiatives stall. Teams mistake activity for impact and momentum for traction.

2. The Dependency Trap Partnerships in AI usually begin with vision and confidence. Everyone is excited, everyone is aligned, and the future is described in bold language.

Then, the drift happens.

It starts when decisions stop sticking. Ownership becomes blurry. Timelines slide without consequence. The hardest choices get deferred behind vague "alignment" meetings.

At first, it feels like collaboration. Over time, you realize you are building around someone else's constraints. Instead of gaining leverage, you inherit limitations. Your Support roadmap becomes a negotiation.

When that happens, you didn't really choose a partner. You chose a dependency. And dependencies always come with a hidden cost.

3. Confidence Without Competence Onboarding is where the real damage happens.

When AI onboarding is done well, it teaches a mental model. It explains what "good" looks like and makes the limits visible. When it is done poorly, you aren't onboarding users. You are onboarding confusion.

Teams get excited because the first demo works. Users walk away believing the AI is reliable in situations where it isn't. They don't understand why it breaks or what to do when it does.

This is the silent killer. The AI doesn't fail loudly; it fails quietly. It produces plausible answers that are wrong and confident outputs that are brittle.

The "scaling tax" here isn't more tickets. It is rework, manual checks, and internal escalation. You end up hiring people not to scale AI, but to supervise it.

4. The Miracle Problem There is a deceptive warning sign that shows up in AI more than anywhere else: early success that can't be explained.

A demo lands. A customer says "wow." The organization wants to treat that moment as validation.

But reality is more demanding than a demo environment. The biggest gap appears when the team can't articulate *why* something works. In those situations, the excitement becomes louder than the evidence. You start building narratives instead of systems.

The risk is not that AI is hard. The risk is that teams build foundations on unpredictability and get surprised when scale requires control.

5. Confusion is a Signal What I've learned is that confusion is rarely random. It is a signal. It tells you exactly where the system is weak.

It shows you where a product isn't teaching, where a partner isn't aligned, and where a launch isn't landing.

Most teams treat confusion like a temporary phase—something that will fade once the technology matures. That is the "good enough" trap. I can't stand it. In reality, confusion compounding is one of the main ways companies lose. Slowly.

Surprised? Why? None of this is pessimism. It's pattern recognition.

The warning signs aren't there to scare us. They are there to give us time to choose differently while the cost of change is still low.

I keep these signals close to my heart as we explore what AI can really do for our teams.

Beware: I’m a tough customer to satisfy.

Field note: *AI is only useful after you can name the operational gap it is supposed to close.*

3. Data Gathering: Building the Case Beyond "Vibes"

Opinions are interesting. Data is actionable.

There is a specific seduction in the world of AI right now. It goes like this: *“Everything is on fire. Let’s buy an AI bot, turn it on, and go to happy hour.”*

That is a great way to burn a pile of cash and annoy your customers.

In the last article, I listed the warning signs of AI adoption. To avoid those traps, we realized that before we bought a single license or made a single API call, we had to prove the model. We had to move from “vibes” to cold, hard audits.

We realized we didn’t need to look elsewhere for answers. We have years of history and hundreds of thousands of past interactions. We didn’t need to guess; we needed to “consult” our own archives.

The Great Audit We opened the hood on our ticket volume. We didn’t just look at the *number* of tickets; we looked at the *soul* of the tickets.

We categorized them into two specific buckets:

1. The Solvable (Wrongly called “Deflectable”) For these, we didn’t just ask “can an AI answer this?” We identified the root cause of the question:

- **The Solution Path:** Does the answer actually exist?
- **The Discovery Gap:** Could the customer have solved this themselves, but didn’t know how?
- **The Developer Experience:** Was the path clear when onboarding a new functionality, like in Sentry Logs (https://sentry.io/product/logs)?
- **The Knowledge Gap:** Are the docs up to date? Does searching our Knowledgebase actually work?
- **The Product Gap:** Is the customer asking for a piece of data or action that *could be* made available, but just wasn’t prioritized yet?

2. High-Touch (The Real Work) These are the tickets where humans shine.

- Is the customer experiencing a complex instrumentation issue that requires deep context?
- For example: is this a stack trace debugging issue specific to an environment like our Unity/Game Consoles SDK, and does it actually require a code change? https://docs.sentry.io/platforms/unity/game-consoles/

The Grunt Work Here is the part nobody puts in the case study because it isn’t sexy: **We tagged thousands (and thousands) of conversations.**

Yes. Manually.

Like that:

As part of our day-to-day, we added a step that enabled us to re-learn what developers ask us, what their pain is, and what the path to resolution looks like.

Here is the field we learned to appreciate:

We spent long hours tagging thousands of tickets. It felt like the opposite of ‘high-tech.’ But you can’t automate a process you don’t actually understand at the atomic level. It was absolutely necessary.

Evidence > Intuition The result wasn’t a hunch. It was a spreadsheet with undeniable gravity.

The data showed that a **significant** percentage of our volume was purely transactional—questions that didn’t require empathy or deep debugging, just accurate information retrieval.

We proved the ROI model before we spent a dime. We built the case on the floor of the warehouse, not in the ivory tower.

Now, we had permission to build. I mean: to buy.

Field note: *Before buying or building, audit the queue. The archive already knows what your customers cannot find, understand, or do.*

4. "We Already Have AI" and Other Logical Traps

There is a specific kind of pushback you get in engineering-led companies.

It isn't political. It isn't emotional. It is brutally, intellectually honest.

When you propose a new tool to a room full of senior engineers and product leaders, you don't get "No." You get: *"Why can't we just use the thing we built last Hackweek?"*

We have the data (see previous Article). We know the pain is real. But now we have to survive the **Meritocracy of Ideas**.

We will need to answer the questions that sound reasonable on the surface but may kill projects in the crib:

The "We Already Have AI" Trap The loudest objection is the most logical one: *"We already have an AI search bar on our docs site. Why do we need to have one for any other flow?"*

The temptation here is strong. If you have a vector database and an LLM, it feels like you have a solution.

But that creates a dangerous confusion. **It confuses a Library Index with a Librarian.**

Retrieval vs. Reasoning vs. Problem-Solving We need to draw a hard line in the sand between three very different operations:

1. **Retrieval (Docs Search):** This is for the user who knows what they are looking for but doesn't know where it is.

- *User:*"How do I configure sampling?"
- *AI Search:*"Here is the link to the Sampling Options page."
- *Success Metric:* Accuracy of retrieval.

2. **Reasoning (Support):** This is for the user whose code is broken and they can't tell why.

- *User:*"My sampling is set to 100% but I'm dropping events on Vercel."
- *AI Support:*"That configuration looks correct, but looking at your SDK version, there is a known conflict with the Vercel adapter in version 7.x. You need to upgrade."
- *Success Metric:* WE got the customer to their final step before a resolution.

3. **Problem-Solving (Solutioning)**: This is for the user who knows what they want to achieve, but doesn't know the best way to implement it (or maybe they simply can't).

- User: "I want to reduce noise. How do I only get alerted to real issues?"
- AI Solution: "Let me turn on alert rules for new issues only, mute known noisy ones for you, and route P0 only issues to the PagerDuty you enabled in your account."
- Success Metric: Job done.

Docs Search throws a book at you. Support reads the book for you, looks at your specific mess, and tells you which page fixes it. This feels like the year 2016.

The "We Can Build It in a Weekend" Trap This is the classic Engineer's Fallacy.

“It’s just an API wrapper around OpenAI. Give me two engineers and a week.”

They aren’t wrong. They *could* build the prototype in a week. But we don’t want to build a prototype. We want to run a full support operation.

We have to gently remind our own team: **We are an Observability Company, not a Support AI Company.**

Every hour our engineers spend maintaining a custom RAG pipeline, debugging context windows, or building feedback UIs for support agents is an hour they *aren’t* building Sentry.

We need to decide to buy leverage, not build technical debt.

The “Is It Safe?” Trap This isn’t an objection; it is a requirement. Sentry holds the keys to the castle for countless companies (read: it’s growing too fast for me to type the actual number, so have a look here: https://sentry.io/customers/). We process their errors. We see their code.

The pushback here is different. It is quiet, serious, and non-negotiable. If the tool hallucinated an answer, that’s bad. If the tool hallucinated customer data into a chat window, that’s catastrophic.

We can’t just “trust” the model. We need a vendor that treats security like a feature, not a footer on their website.

The Turning Point The internal conversations are intense. They are detailed. But they are positive. The team isn’t trying to kill the project; they are trying to stress-test the logic.

Once we can prove that:

4. Search is not a Support function.
5. Building it ourselves is a distraction.
6. The security model holds water.

The mood will shift.

The skepticism will turn into curiosity and the objections will stop being walls and start being requirements.

We have the data. We will have the buy-in. We will have the budget.

On paper, this could work!

I had a moment of reflection.

As I look at the landscape of vendors - the “Hot New AI Startups” promising to solve all our problems - I realize we are about to walk into a minefield.

Winning the internal argument will be the easy part. Next we will need to find a tool that actually works.

Field note: *Having AI somewhere in the stack is not the same as changing the customer experience.*

5. The Goalkeeper Fallacy

Traditional Support waits for the customer to fail. We decided to intercept them before they do.

The Death of the Safety Net

In the world of SaaS, most Support organizations are built like world-class Goalkeepers. We pride ourselves on the "Great Save." We celebrate the Support Engineer who dives into a 4:00 PM Friday fire, deflects a catastrophic bug, and earns a "Thank You" note from a frantic CTO.

It's heroic. It's emotional. And it's a fundamental strategic failure.

If your goalie is the busiest person on the pitch, you aren't winning; you're surviving. You are playing a defensive game, pinned in your own box, waiting for the customer to make a mistake, break the product, or lose their patience. By the time a customer "submits a ticket," the friction has already occurred. The goal has been scored against you. You're just cleaning up the net.

From Goalie to Midfielder

We've spent the last year burning the "Goalkeeper" playbook. We are moving our defensive line up thirty yards. We are moving into the **Midfield**.

In soccer, the midfielder is the playmaker. They don't wait for the ball to reach the goal line; they intercept the pass. They read the game, disrupt the opponent's flow, and transition immediately from defense to attack.

When we talk about "Moving Support to the Midfield," we are talking about **Proactive Interception**.

The traditional triage reality—where a user hits a snag, hunts for a "Contact Us" link, fills out a five-field form (argh), and waits for an asynchronous and never relevant "we've received your request" email—is an obsolete model. In fact, it's worse than obsolete; it's a deterrent. It actively punishes the user for needing help.

The 90% Silent Churn

CROs often look at CSAT and think, "Our customers love us!" because the people who write in are happy with the fix. This is a survivor bias trap.

The real danger isn't the 10% of users who saw something and opened a ticket. The danger is the 90% who saw the same thing and did not. These are the prospects who hit a friction point, can't find an immediate answer, and—rather than waiting 4 to 24 hours for a human response—simply close the tab.

They don't complain. They just leave. They are the "Lost" customers, and your Goalie never even saw the ball.

The Fix: Interception

Our strategy document was clear: if the traffic starts in the product, the solution must live in that same UI.

By embedding an AI agent directly into the user's workflow—not buried in a Help Center three clicks away—we changed the physics of the interaction. We stopped asking the customer to come to us. We went to them.

When a developer is staring at a 401 Unauthorized error in the console, they don't want a "relationship" with a support agent. They want a solution, and they want it at the speed of Wispr. By putting AI in the "Midfield"—intercepting that error message before it turns into a "Contact Support" thought—we capture that 90% of silent traffic.

We aren't just "deflecting tickets." We are protecting the user's momentum.

The Impact on the P&L

For a B2D2B (Business → Developer → Business) SaaS company, the Midfield model is the only way to decouple headcount from revenue. If you stay in the Goalkeeper model, your costs scale linearly with your problems. More customers = more tickets = more goalies.

By moving to the Midfield, you are investing in a system that scales through **intelligence, not empathy.** You save your humans for the "Championship" moments—for the high-value architectural consultations—and let the AI play the Midfield, intercepting the thousands of daily "passes" that used to end up as goals against your team's productivity.

It's time to stop celebrating the "Great Save" and start celebrating the "Interception."

Field note: *The support team should not only save shots on goal. It should move upfield and intercept confusion before it becomes a ticket.*

PART II

THE MARKET AND THE MINEFIELD

How to choose AI support without buying garbage or outsourcing judgment.

Once a company admits the friction is real, the market arrives with a thousand confident answers.

Everyone is an AI company now. Every vendor has a demo. Every deck promises resolution, deflection, transformation, and scale. The hard part is not finding AI. The hard part is finding a system that respects your product, your customers, your security posture, your escalation philosophy, and your tolerance for embarrassment.

This part is the minefield. It is where the wizard, often dressed as a polished founder, hides the humans behind the curtain. It is where Sales disguises a rent-seeking tollbooth as strategy. And it is where Product may be right: sometimes the best answer is to build.

6. The Landscape: Everyone Is an AI Company Now

The sun came up. The coffee is brewing. The adrenaline from getting the final word in the “Build vs. Buy” conversation has faded.

Now I have a terrifying realization: The dog caught the car.

We successfully convinced the organization that we shouldn’t build this ourselves. We argued that “buying leverage” is smarter than “building debt.” The leadership agreed. We have the budget. We have the mandate.

Now, we actually have to find a tool that works.

I opened my inbox this morning. I searched for “AI.”

I shouldn’t have done that.

The Gold Rush

If you are a GTM leader, you know the sensation. Every SaaS tool you have ever touched, browsed, or accidentally clicked on is now an “AI Platform.”

Your CRM has AI. Your ticketing system has an AI add-on. The tool you use to schedule tweets (guilty) has AI.

There are currently two types of companies in the valley:

7. **The Incumbents:** Massive platforms bolting a chat interface onto legacy code and calling it “Copilot.”
8. **The Wrappers:**Three engineers in a WeWork with a cool domain name, a dark-mode website, and a thin UI layer over GPT-4.

We are walking into a minefield.
We are looking for the third kind.

The Problem with “Generic” AI

Here is the specific challenge at Sentry: Our customers are developers.

They aren’t asking “Where is my refund?” or “How do I change my password?”

They are asking: *“My Python SDK is dropping events when I use the Celery integration with sampling rate 0.5. Why?”*

If we put a generic support bot in front of a Sentry user, we won’t just annoy them. We will insult them.

We need a tool that doesn’t just “chat.” We need a tool that understands documentation, reads code snippets, and understands the difference between a “bug” and a “configuration error.”

We need a Librarian, not a parrot.

The Three Mines

As we look at the vendor list, we are looking for three specific things that will kill a deal immediately.

1. The "Wrapper" Mine We just spent Article IV convincing our engineers *not* to build this because it's a distraction. I cannot go back to those engineers and say, "Hey, I bought this tool, It's basically exactly what you said you could build in a weekend." If the vendor adds no value beyond the raw model - if they don't have better indexing, better retrieval, or better connectors - we aren't buying it.

2. The Security Mine Sentry holds the keys to the kingdom for our customers. We process their crashes. I cannot have a vendor that trains their public model on our private data. "Trust us, we promise," is not a security policy. We need SOC2. We need data isolation. We need a vendor who understands that "Enterprise Ready" means more than just having a "Contact Sales" button.

3. The "Agent" Mine This is the loudest buzzword right now. "Autonomous Agents". The promise that the AI will go into your system, click the buttons, fix the bug, and email the customer. I love ambition. But I don't trust a toddler with a chainsaw, and I don't trust a V1 AI agent with write-access to my production code. For now, we want a **Co-pilot**, not an **Auto-pilot**.

The Selection Process

We are going to do this the old-fashioned way.

We aren't going to look at the marketing videos. We aren't going to look at the venture capital funding announcements.

We are going to take the hardest, messiest, most technical issues that hit our inbox last week - the ones that made our senior support engineers sweat - and we are going to feed them into these tools.

If they hallucinate? They are out. If they give generic advice? They are out. If they tell us to restart the router? They are banned.

Winning the internal argument was the easy part. Now we have to find a partner who is actually building the future, not just marketing it.

Let the bake-off begin.

Field note: *The AI gold rush makes every vendor sound inevitable. Your job is to separate demos from systems.*

7. The Wizard of Oz

In the advanced technology ecosystem, progress is driven by two separate yet equally important groups: ***the innovators who build groundbreaking AI***, and ***the vendors who exploit the hype with hollow promises***. These are their stories. DUN-DUN.

The Holy Grail.

We thought we found it. In the middle of our vendor search, we found a platform that didn't just promise "co-piloting." They promised full Autopilot. 99% deflection. Zero human touch.

The pitch was incredible. We threw complex Python stack traces at their sales team. They showed us demos where it solved them. We threw vague billing questions at it. It answered them.

It was slick. It was fast. It was expensive.

The "Onboarding" Red Flag:

We were ready to move to the Proof of Concept (POC). We expected them to ask for API keys, documentation links, or a dump of our redacted tickets to "fine-tune" their model.

Instead, they sent us a spreadsheet.

It was a questionnaire. And looking at the questions, the mask slipped.

- "What is your monthly ticket volume?"
- "What are your Top 5 most common issue types?"
- "Please list the exact step-by-step resolution for your Top 5 issues."
- "What are the most common follow-up questions for billing inquiries?"

I stared at the screen. Why does an advanced LLM need me to manually list my top 5 issues?

If this were true Generative AI, it would ingest our 10,000 public documentation pages and our last year of ticket history and tell us what our top issues are. It would figure out the resolution by reading the docs, not by asking me to type it into a cell in Column C.

The Ghost of GPT-2 Past

It immediately clicked. I felt like I was back in 2019.

For those who remember the first wave of "AI Customer Support" (the GPT-2 era), this was the standard playbook. Back then, the tech wasn't good enough to actually think. So, startups weren't selling software; they were selling tech-enabled services.

You would send them Excel files and decision trees. They would take those files and hire a massive back-office workforce—usually in a low-cost geography—to manually act as the "intelligence."

When you chatted with the “bot”, you were actually chatting with a human who was frantically CTRL+F’ing through the spreadsheet you filled out during onboarding.

The Mechanical Turk

That’s why they needed the “Top 5 Issues.” They weren’t fine-tuning a neural network; they were writing a script for a call center agent.

They needed to know the volume so they knew how many humans to hire for the shift. They needed the “Standard Resolutions” so they could tape a cheat sheet to the agent’s monitor.

They weren’t selling us a brain. They were selling us a BPO (Business Process Outsourcing) team wearing a “Generative AI” mask.

The “Values” Check

Let’s call it what it is: Deception.

This wasn’t just a product failure; it was a security nightmare. We are Sentry. We process sensitive data. We value Security, Privacy, Compliance and Resilience (https://sentry.io/trust/) and we thought we were going to pipe some data into a SOC2-compliant, stateless model. In reality, we were expected to handle that data to an unknown human, in an unknown location, with unknown security protocols, armed with a spreadsheet we wrote for them.

The Lesson: Verify

The AI market is currently a gold rush, and where there is gold, there are snake oil salesmen.

At Sentry, one of our core values is “Pixels Matter” - we understand that the difference between a good product and a great one is in the finer details. We take extra care to get everything right, down to the last pixel. We build for developers. Developers are the most skeptical audience on earth. If we shipped a “support bot” that was actually just a hidden human reading a script, our community would find out in a week. They would tear us apart on Hacker News, and they would be right to do so.

We don’t want magic. We want perfect engineering.

New rule, ignore emails with meaningless subject lines (“AI Delivers on CX, and It’s Driving Real Revenue”).

I was invited to a black-tie gala with their CEO. I said I’d rather build.

Back to work. We have more vendors to test.

Field note: *If the AI works only because hidden humans are carrying it, you have not automated support. You have obscured labor.*

8. The Security Gauntlet

Five years ago, a vendor security review was a polite dance. We sent them a questionnaire. They sent us a SOC2 report. We checked a box. Everyone went to lunch early.

Today, the conversation is different. It is shorter, sharper, and much more paranoid.

When we evaluate an AI tool in 2025, we don't start with "Do you encrypt data?". We start with the question that actually matters: **"Who is training on my data?"**

We Know Because We Build It

We aren't asking these questions because we are luddites. We are asking because we are builders.

At Sentry, we built an AI Debugger that uses Sentry context, not just tells you what broke; it opens a Pull Request to fix it. To make that work, we have to process your code, your stack traces, and your error logic.

We know exactly how tempting it is to say, *"Hey, if we just trained on everyone's data, the model would be 10% smarter."* We also know that if we did that, we would lose the trust of every developer on the planet overnight (see https://docs.sentry.io/product/ai-in-sentry/ai-privacy-and-security/).

We know that trust is binary: you either have it, or you don't.

So when a vendor tells us, "Trust us, it's safe," we don't accept it. We dig.

Few Example Questions

We created a specific "AI Addendum" to our security review. It has at least two non-negotiable questions.

1. The Training Question

- **The Ask:** "Does my data improve the model for your other customers?"
- **The Trap:** Vendors will say "We use data to improve the service."
- **The Reality:** "Improving the service" is legal-speak for "We train on your data."
- **The Requirement:** We need a hard, contractual guarantee that our data is sandboxed. If Sentry code teaches your model how to write Python, and then that model writes Sentry-style Python for a competitor? **Hard Pass.**

2. The "Un-learn" Question

- **The Ask:** "If we churn tomorrow, how do you remove our influence from the model?"
- **The Trap:** "We will delete your account."
- **The Reality:** You can't "delete" data from a trained neural net. You can't Lobotomize an LLM.
- **The Requirement:** Since you can't un-train a model, we strictly require that you never trained on it in the first place (see Question 1).

The New Operating Order

Here is the good news: The high-quality vendors *love* this.

The “WeWork Wrappers” crumble under this scrutiny. They don’t have Zero Data Retention policies. They rely on (e.g.) OpenAI’s default settings. They can’t sign the DPA.

But the serious players? They have a “Trust Portal” ready. They have the architecture diagrams showing the isolation between the Vector DB (long-term memory) and the LLM (reasoning engine).

We found that the best vendors aren’t hiding their architecture. They are flaunting it. They know that in the enterprise, **Privacy is the Product.**

The Verdict

We put our final candidates through the Gauntlet. They didn’t just pass; they showed us how they write the rules books for everybody else.

They have the features. They have the security. We are out of excuses.

It’s time to turn it on.

Field note: *Trust is not a blocker to AI support. Trust is the design constraint that makes AI support deployable.*

9. The Paper Trail of a Vision

In Silicon Valley, we like to pretend that “vision” is a lightning bolt: a single moment of clarity that changes everything. In reality, vision is a mountain of paperwork. It is the unglamorous, caffeinated work of writing the same story ten different times, in ten different languages, until everyone from the CRO to the Support Engineer sees the same future.

We didn’t just need a new tool. We needed a **Translation Layer.**

The Sci-Fi Doc: Visualizing the Inevitable

The first thing I created wasn’t a requirements doc. It was a “Sci-Fi Vision.”

I didn’t want to talk about API endpoints or ticket routing. I wanted to describe a random Tuesday in 2027. I wrote a narrative about a Support Engineer who walks into their shift and, instead of staring at a queue of 40 “How do I...” tickets, sees a predictive Command Center.

To make it visceral (AI word), I used **Gemini** (nano banana) to help imagine this future. I didn’t want wireframes that looked like a 2015 Jira; I wanted high-fidelity “screenshots” of a predictive HUD that surfaced the fix before the customer even finished typing. When the team saw that UI, the conversation shifted from *“Is this possible?”*to *“When do we get this?”*

The Audience: Everyone.

The Purpose: To make the “impossible” feel “inevitable.”

The Translation Layer: Bridging the Great Divides

A document is just paper. A Translation Layer is a tool. I realized everyone was looking at the same problem but seeing a different threat. My job was to build the bridge between four distinct worlds:

- **The Trenches vs. The Tower:** There is a massive gap between those in the trenches—feeling the heat of every ticket—and those in the tower who “think they know” because they look at a green dashboard once a week or randomly read a support question on X. I turned the raw, bloody reality of a support shift into data that the “tower” couldn’t ignore.
- **Career Aspirations vs. The New Economy:** Let’s be real—the 2026 economy is different. My team doesn’t want to be “Human Knowledge Bases” forever. They want to be **System Architects.** I had to translate our new tools as a career accelerator. We aren’t replacing the team; we are replacing the *boring parts* of their jobs.
- **The Metrics Lie vs. Customer Success:** Our current metrics are “Good Enough” traps. They measure speed (of all sorts), not resolution. I wrote the new metrics doc to tie Support directly to company growth. We moved from “How many tickets did we close?” to **“How much friction did we eliminate?”**
- **The DX Gut Check vs. Our Brand:** Sentry is a brand built for developers. We pride ourselves on elite Developer Experience (DX). But our internal and support tools? They were a mess of 50 (AI number) tabs.

I had to ask loudly: **“Does this clunky experience fit our brand?”**If we wouldn’t sell it, why are we forcing our peers to use it?

The Brutal Reality: Gap Analysis & The Theoretical Demo

Then came the cold shower: The Gap Analysis. This was a technical autopsy of our current architecture versus the 2026 volume.

I created a **Theoretical Demo**—a side-by-side comparison of the “Now” and the “Future.” I didn’t just show slides; I showed the friction. I highlighted the ghouls in our system—the 2015 tools snapping under the weight of our elite product velocity. It could have shown Product exactly if and why new features were being “copy-pasted” into oblivion instead of being supported.

The Human Echo: Mining the Pain

To ensure I wasn’t dreaming in a vacuum, I interviewed our customers and learned from the successes of others. I fed those raw, messy interview transcripts into **NotebookLM.** I’m an ESL writer, and I needed NotebookLM to be my “Story Architect.” I asked: *“Where is the pain the loudest?”*and *“How do I say this so I sound like a peer, not a manual?”* It helped me bridge the gap between my technical intent and the emotional reality of a frustrated developer. It turned data into a **“Before & After” Doc** that proved every “missed experience” was a direct hit to our reputation.

The Strategic Hammer

The buying phase turned everything into ROI: comparisons, budget lines, and business-case math. But the truth was clear from the beginning: this wasn’t a typical software purchase. It was a way to scale without pretending more headcount could solve a physics problem.

The Choice: Moving or Crashing

Throughout this entire paper trail—from the Sci-Fi vision to an RFP—there was one underlying truth I made everyone face:

Choosing not to move forward is a choice to stay idle. And in this market, staying idle is choosing to crash.

We often frame “doing nothing” as the safe, conservative play. It isn’t. In the face of rising volume and elite technical velocity, doing nothing is an active decision to let the wave crash over our heads. We didn’t just choose a new tool; we chose to stop being victims of our own growth.

Field note: *One AI strategy has to speak many organizational languages: product, security, support, legal, GTM, finance, and engineering.*

10. The Product Veto

In most companies, the Support team buys software in a silo. They find a tool that makes ticket routing 5% faster, they buy it, and the Product team ignores it.

But Sentry is not “most companies.” And we aren’t selling socks. We are selling observability to developers.

If our product interface is a sleek, dark-mode, high-performance Ferrari, but our Support interface looks and acts like a 1998 Honda Civic with a leaky radiator, we have broken the spell. And that was the limit of my understanding of vehicles.

We didn’t just ask IT to approve the vendor. We asked the people who built Sentry.

The Guardians of DX

Our Product Managers and Engineers are obsessively protective of the **Developer Experience (DX)**. They determine how the product feels, how the APIs respond, and how the docs are structured.

Usually, they view Support tools as “Enterprise Bloatware” - necessary evils that they hope they never have to log into.

We dragged them into the room and said: *“This AI is going to talk to your users. It is going to represent your code. If it sucks, it’s on you too.”*

The Litmus Test: “Does it Speak Code?”

The feedback from the Product team wasn’t about ROI or deflection rates. It was about **respect for the craft.**

When we showed them the shortlist of vendors, they didn’t look at the analytics dashboard. They looked at the chat window.

The Engineer’s Potential Objection: *“Look at this,”*a Senior Frontend Engineer may say during a demo. *“The AI pasted a Python snippet, but it stripped the indentation and it isn’t using syntax highlighting. It’s unreadable.”*

To a normal buyer, that’s a nitpick. To a developer tool company, that is a dealbreaker.

The PM’s Potential Ultimatum: *“This bot is too polite,”*a Product Lead may say. *“It sounds like a bank teller. Sentry isn’t a bank. We are direct. If the user messed up the config, tell them they messed up the config. Don’t apologize for their syntax error.”*

Extending the Surface Area

The Product team is helping us realize something critical: **DX doesn’t stop at the sentry.io domain.**

We meet customers in the IDE. We meet them in the CLI. We meet them on GitHub. And now, we meet them in this chat window.

If the AI acts like a generic “Support Agent” it creates a jarring disconnect. If the AI acts like an “Engineer”: smart, context-aware, and technical - it feels like an extension of the product.

The “No Garbage” Rule

The Product team prioritize a vendor that:

9. **Understood Technical Context.**
10. **Allowed “Agent Voice” Customization.**

The Alignment

Support and Product were looking at the same tool with the same goal. Support wanted **leverage** (help customers help themselves), Product wanted **consistency** (support the brand).

We are THIS close to finding a tool that did both. It respects the code. It respects the user. And most importantly, it respects dark mode (hey). My Terminal theme may raise questions, but it also builds character.

We got the team’s blessing.

Field note: *Do not buy garbage. If the tool cannot respect your product, your developers, and your edge cases, it will burn trust faster than it saves time.*

PART III

THE MACHINE

Architecture, read/write risk, metrics, and the support engineer's new job.

A support machine is not a prompt. It is an architecture of judgment.

The machine needs memory, retrieval, reasoning, routing, action, measurement, and restraint. It needs to know when to answer, when to ask, when to act, and when to stop. The moment it crosses from read-only to read/write, the Rubicon is no longer a metaphor. It is a safety boundary.

This part turns the story from adoption into design. If Part II asks what to buy, Part III asks what must be true for the machine to deserve power.

11. My Dream AI Architecture

The Blueprint - Architecting a "Separation of Powers"

When people think about "AI Architecture," they usually imagine a giant brain sitting in the middle of a server room. But if you look at modern **AI Blueprints**, you realize that a good AI architecture isn't a "brain" - it's an **orchestra.**

The Separation of Powers

The biggest takeaway from modern AI blueprints is the move away from the "One Big LLM" approach. Instead, we are seeing a rigorous **Separation of Powers**:

11. **The Librarian (Retrieval):** This layer doesn't "think"; it just finds. It's the RAG (Retrieval-Augmented Generation) pipeline that indexes our knowledge and past tickets. Its job is purely to provide the "Grounded Truth."
12. **The Judge (Reasoning):** This is the LLM. It takes the user's mess and the Librarian's facts and decides what is relevant. It doesn't act; it only evaluates.
13. **The Clerk (Execution):** This is the Agentic layer. Once the Judge has a plan, the Clerk executes it.
14. **The Gatekeeper (Security):** This layer inspects. It is the security checkpoint that sanitizes inputs and blocks the wrong type of outputs before the model ever sees them. Its job is purely to enforce the safety laws of the system.
15. **The Auditor (Evaluation):** This layer critiques and grades the work done by the Clerks, before the user sees it. Its job is purely to catch hallucinations and force a retry if the standard isn't met.

I ran every model through my "The Mashed Potato" Test. I simply ask: 'What goes best with mashed potatoes?' The internet consensus is 'gravy,' but the ultimate truth is 'sour cream.' I use this to see if the AI defaults to General Consensus (its training), finds the Absolute Truth (my context), or has the discipline to realize it should not answer this one at all.

By separating these roles, you avoid the "Miracle Problem" I mentioned in previous articles. If the AI gives a wrong answer, you don't have to retrain a massive model; you just check if the Librarian gave the Judge the wrong book.

What the Architects are Asking

While the "Separation of Powers" keeps the system stable, regular software architects -the ones choosing their AI partners- are looking at a much colder checklist. They aren't seduced by the "vibes" of a demo; they're looking for:

- **Dependency Governance:** "If we pick this AI partner, are we locked into their model forever?" Architects want a **Multi-Model Strategy** where they can swap GPT for Claude (or an internal Llama-4 instance) without rewriting the entire support engine.

- **Cost Observability:** In 2026, “Token Spend” is the new “AWS Bill.” Architects need to know the cost-per-resolution. They want to see that the system uses a $0.001 model for simple triage and only calls the $0.10 “genius” model when a complex stack trace appears. Even better, they want to never worry about it and never ask to swipe their credit card in order to ‘resume operation’.
- **The “I Don’t Know” SLA:** A professional architect values a system that admits it’s lost. They look for **Confidence Gating**—the ability for the AI to hit a “Human Escalation” trigger the moment the data retrieval score drops below 85%.
- **Data Residency & Privacy:** Can we run the Librarian layer on our own VPC? Architects want the leverage of AI without the risk of their proprietary codebases leaking into a public training set.

Moving Beyond the Bot

What our *future vendor blueprint* will provide is a roadmap to stop building “features” and start building a **Knowledge Engine.**It will validate that our manual tagging (Article 3) wasn’t just busywork - it was the essential process of defining the “Constitutional Law” our AI must follow.

We aren’t just adopting a bot to answer tickets. We are building a system that knows when to speak, when to act, and—most importantly—when to stay silent and call for a human.

Field note: *The machine needs separation of powers: retrieval, reasoning, action, governance, and human judgment cannot all blur into one prompt.*

12. The Read-Write Rubicon

Answering questions is cute. Fixing [account management issues] via API is profitable. Here is why we want to give the robot the keys (safely). One day. Soon.

Crossing the Rubicon

In the early days of the AI boom, we were all playing in a sandbox. We deployed "Read-Only" bots—like our early work with Kapa.ai—that were essentially glorified librarians. They could read your documentation, summarize a paragraph, and point a user to a link.

It was helpful. It was "cute." But it didn't move the needle on the bottom line because it didn't actually *do* anything.

The real shift—the "Read-Write" Rubicon—happens when you stop treating AI as a search bar and start treating it as an employee with permissions. Crossing this river means moving from "Here is how you increase your pay-as-you-go allocation" to "I have updated your subscription for you."

One is a suggestion; the other is a transaction. And in SaaS, transactions are the only things that scale.

The "Human API" Problem

We have to be honest with ourselves: for the last decade, we have been hiring brilliant Engineers and turning some of their day to day into "Human APIs."

When a customer wants to set a new release version and feels stuck, they open a ticket. A human reads that ticket, logs into a back-office admin tool, clicks three buttons, and emails the customer back with new information.

That is a $???k-a-year engineer acting as a slow, manual interface for a task the customer should have been able to do themselves. It's a waste of human intellect and a bottleneck for customer velocity. We weren't providing "white-glove service"; we were providing a high-latency API call with a human face. We were not the "Zappos of Application Monitoring software".

The Execution: Self-Healing Support

To move our operation from "Read-Only" to "Read-Write," we have to build the plumbing. We will map low-risk, high-frequency actions—things like subscription checks, and finding DSN —directly to **our new Agent Actions** and **Data Connectors**.

This is the birth of **Self-Healing Support**.

Soon, when a user asks our Agent about their budget allocation, it will not just quote the pricing page. It queries the billing API, identifies the budget, allocation, and—within predefined guardrails—offers to process

to make changes to it. The “Human API” is bypassed. The customer gets a resolution in 60 seconds instead of 6 hours.

The Safety Net: Determinism over Improvisation

The biggest fear for any CRO or CTO when “giving the robot the keys” is the hallucination. No one wants an AI to “improvise” a 100% discount for every customer because it was trying to be “helpful.”

We will solve this through **Deterministic Controls**.

When our Agent interacts with our backend, it isn’t “thinking” its way through the database. It is triggering specific, hard-coded workflows. We use the LLM to understand the *intent*(the “Read”), but we use standard, rigid code to execute the *action*(the “Write”).

The AI can’t rewrite the logic of a refund; it can only call the process_refund function if the conditions we’ve set are met. We didn’t give the robot the keys to the house; we gave it a very specific set of smart-locks that only open when the right criteria are hit.

The Profitability of Action

The “Read-Write” shift is where the ROI of AI becomes undeniable. Every time our Agent “pushes the button” so a human doesn’t have to, our customers and team are happier . We are no longer paying for the *execution of* a task; we are paying for the *architecture that* allows the task to execute itself.

We’ve stopped answering questions. We’ve started solving problems.

Field note: *Read-only AI is assistance. Read/write AI is power. Crossing that line changes the safety model.*

13. The Support-to-GTM Feedback Loop

In too many organizations, Support is often the unsung hero, the quiet engine room keeping the ship afloat while others take the credit for the speed. But as a company matures in its understanding of the customer, a fundamental shift occurs: Support moves from the "back office" to the very front of the value chain. The more a company truly respects its users, the more it realizes that the Support team isn't just a safety net—it is the ultimate guardian of the customer's trust. When a company reaches this level of sophistication, they stop viewing Support as a cost to be managed and start seeing it as the most direct, high-impact touchpoint in the entire GTM motion.

Watch me tell it like it is without saying 'Forward Engineer Support Team' even once.

At this level of alignment, the "No Garbage" rule becomes a badge of honor for the Support team. It's a recognition that if we are selling a premium, developer-centric product, the Support experience must be the highest expression of that craft. The GTM team realizes that their sales and expansion goals are directly tied to how Support handles the "moment of truth"—that critical window when a user is stuck. If the GTM motion is built on a "Ferrari" promise, it is the Support team that ensures the engine never fails. They aren't just resolving tickets; they are protecting the integrity of the brand. In this model, the Support team doesn't just "get a seat at the table"—they are the ones defining the table's dimensions.

This synergy creates a powerful feedback loop where Support and GTM goals become indistinguishable (<-AI word). Support aligns with the customer journey by understanding that a self-serve user needs the friction-less precision of a "smart" technical interface, while a sales-led enterprise account needs the nuanced, authoritative partnership of a technical expert. Because Support has the deepest insight into where the product succeeds and where it stumbles, they become the strategic advisors to the Product and Sales teams. They aren't just fixing what's broken; they are actively shaping the roadmap to ensure the "Ferrari" stays ahead of the competition.

I don't like cars. I don't really care for them. But I assume you do.

Ultimately, when a company values its customers, it provides Support with the best possible people and tools because anything less is an insult to the user. Support takes the "Veto rights" because we believe the Support experience is so vital to our success that it deserves the same obsessive engineering rigor as the core product itself. When Support, GTM, and Product align, they create a seamless, high-fidelity (<- another AI word) experience that respects the user's intelligence and time. This isn't just about efficiency - it's about building a company that operates with radical transparency and systematic excellence, where the Support team is recognized as the essential architects of long-term customer loyalty.

I'm biased, but for all the right reasons.

Field note: *The queue is not just a cost center. It is a market signal stream if you build the loop to hear it.*

14. CSAT and Other Comfort Metrics

Measuring 1% of your angry customers via email survey is 1990s tech. We need to score 100% of conversations with AI.

The Feedback Mirage

For decades, the Customer Satisfaction (CSAT) score has been the “North Star” of Support. We chase the gold star, the “Great” rating, and the 95% satisfaction goal. Yes, the Sentry team is exceptional.

But here is the dirty secret that CS leaders hate to admit: **CSAT is a lie.** It’s a data set built on extremes. CSAT amplifies the loud edges. Most customers never contact support in the first place, so we learn nothing about how they feel. And even when they do, only 3%-10% of them respond. What’s left is a tiny, biased slice pretending to represent the whole business.

When you manage your business based on a 2% actual response rate, you aren’t leading; you’re squinting at a mirage. (←AI-generated phrasing)

The New North Star: CX Score

We retired making the CSAT our main KPI.

Enter the **CX Score**.

This CX (Customer Experience) Score will use AI to analyze most customer conversations for resolution, sentiment, and service quality. We want to reduce reliance on manual surveys. While the exact formula will evolve over time, the approach will surface clear drivers behind the score, such as answer quality or excessive customer effort, allowing us to pinpoint specific areas for improvement rather than relying on abstract metrics. Yes, we are familiar with the once trendy, pre-AI, manual Customer Effort Score (CES).

Instead of waiting for a biased survey, we will task our AI agent to evaluate **conversations** against a standardized rubric. The AI doesn’t care if it’s a Tuesday or if the customer had a bad cup of coffee. It looks for objective markers:

- **Resolution Quality:** Did we actually solve the problem or just “close” the ticket?
- **Sentiment Shift:** Did the user start frustrated and end empowered?
- **Effort Score:** How many hoops did we make them jump through?

By scoring every single interaction, we moved from a “sample size” of X tickets to a total visibility of X^n}). That is the difference between an anecdote and an insight.

How efficient are we?

By shifting our focus to the CX Score, we prove we could grow the business while keeping the team smart and engaged. Because we aren’t measuring human effort anymore; we’re measuring **Systemic Resolution.** When

the AI handles 80% of the volume at a “Grade A” CX Score, you don’t need to hire defenders, **your team are now builders.**

The “RTFM” Metric

We’ve also killed the “Ticket Volume” metric. In the old world, a high ticket volume meant the team was “busy” (which we mistakenly rewarded).

Now, we track the **Automated Resolution Rate.** If a customer asks a question that is clearly answered in our docs, and a human has to paste that link, we consider that a failure of the system. We call it the “RTFM” gap. Every time a human answers a “Googleable” question, it’s a signal that our AI knowledge loop is broken.

We stop rewarding ourselves for “pasting links” and start rewarding us for ensuring the AI never has to ask for that link again.

From Sentiment to Strategy

CSAT tells you how someone *felt* yesterday. The CX Score tells you how your product is *performing* today.

We will start seeing patterns that surveys miss. We don’t just see “support issues”; we see other friction points, documentation gaps, and UI failures in high definition.

We don’t only want your thumbs up / five-star rating. We want zero-effort resolution for our customers.

Field note: *Comfort metrics can hide operational truth. The new scorecard must measure resolution, escalation quality, trust, and learning.*

15. The Promotion

We are promoting the team to manage the machine.

The Elephant in the Zoom

Whenever a CRO hears “AI Automation,” they think of efficiency. Whenever a Support Engineer hears it, they think of extinction.

There is an underlying anxiety in every support org right now: “If the bot handles 80% of the tickets, what happens to me?” If you don’t address this head-on, your team may subconsciously doubt your AI implementation. They’ll find reasons why the bot “isn’t ready” or why “customers hate it.”

We decided to skip the corporate jargon and tell the truth: New roles will shape our industry and **Support System Architect** is the most important one in the years to come.

We aren’t downsizing. We are promoting the entire department.

Working ON the Business, Not FOR the Queue

In the old world, the Support Engineer was a slave to the queue. Success was measured by how fast you could clear the board. It was a hamster wheel—no matter how many tickets you solved today, there would be 500 more waiting for you tomorrow.

In the AI Maturity model I’m acting on, the team moves from “Working FOR the Queue” to “Working ON the Business.”

Instead of answering the same question 50 times, the engineer’s job is to identify *why* the question is being asked, build the logic to automate it once, and then ensure the bot never has to ask for help on that topic again. You are no longer the frontline soldier; you are the General directing the digital infantry.

The New Org Chart

To drive this impact, we need to define new roles that didn’t exist three years ago. If your LinkedIn title still says “Support Specialist,” you’re already behind. Here is how we are restructuring:

- **AI Ops Lead:** The “Conductor.” They monitor the performance of the AI Agent, looking for “hallucination trends” and optimizing the hand-off points between bot and human.
- **Knowledge Manager:** Soon to be the most valued role in SaaS. They don’t just write docs; they curate the “Brain” of the AI. They ensure that every product update is instantly ingested and translated into “AI-ready” semantic data.
- **Conversation Designer:** Part linguist, part UX designer. They craft the “persona” of the AI and design the deterministic workflows that allow the Agent to push buttons safely.

The Knowledge Loop: Solve Once, Scale Forever

The “Promotion” is best summarized by the **Knowledge Loop**.

In the legacy model, knowledge was short-lived. A human solved a problem, the customer was happy, and that knowledge stayed locked in that human’s brain (or a buried Slack thread).

Now, when a complex issue hits a human (the other 20% that the Agent can’t handle), the goal isn’t just to “solve the ticket.” The goal is to **feed the machine.**The human solves the edge case, documents the solution, and “checks it in” to the AI’s knowledge base.

The next time that issue appears? the Agent handles it. You’ve successfully “automated yourself out” of a boring task so you can focus on the next architectural challenge.

The Value Shift

For my CRO, this is a massive win. We are taking our most knowledgeable employees—the people who know exactly where your product operates best, or less so—and moving them into roles where they can actually *fix* those problems at scale.

We aren’t paying for “hours logged in a queue”. We are paying for “systems built.” We’ve started hiring people to build the systems that *empower* customers.

Field note: *The support engineer's job does not disappear. The best version becomes system architecture.*

PART IV

THE MIGRATION

Planning, partner risk, shadow runs, launch inventory, and zero hour.

A migration is where belief becomes operational risk.

The old world has muscle memory. The new world has promise. Running both forever feels safe, but it usually preserves the old physics while adding new complexity. At some point the team has to choose the migration, name the owners, run the shadow test, stock the launch inventory, and stand in the room when zero hour arrives.

This part is the launch arc. It keeps the drama because the drama is real: the machine is about to meet customers.

16. The Hernán Cortés Migration Plan

We will move the “entire Support operation”. Here is why the “Hybrid” model is a trap.

The Point of No Return

In 1519, Hernán Cortés landed in Veracruz with a clear mission. To ensure his men wouldn’t succumb to the temptation of retreat when things got difficult, he didn’t just give a pep talk—he burned the ships. There was no “Plan B.” There was only the forward march. (I asked AI to tell this story in less than 4 sentences).

In the world of Customer Experience transformation, most companies are terrified of new architecture. They try to maintain a “Hybrid” model: they keep their legacy ticketing system and try to “bolt-on” a new AI layer. They want the safety of the old world while flirting with the efficiency of the new.

We realized very quickly: **The “Bolt-On” strategy is a trap.** It creates a fragmented experience for the customer, the internal team and a technical debt nightmare for the team. We decided to burn the ships.

The “Bolt-On” Trap

If you’ve ever tried to retrofit a 1990s engine with a modern electric battery, you know it’s a mess of wires and compromises. (No, I don’t know any of that but AI sure does).

The legacy “Ticket” mindset—pioneered by companies like Zendesk—is built on a foundation of asynchronous, email-based friction. It’s a “store-and-forward” architecture. Trying to layer a real-time AI on top of that feels like a patchwork quilt. You end up with “Franken-stack” Support:

- Data is siloed.
- The AI doesn’t have native access to the UI.
- The customer feels the poor handoff between two different systems.

We rejected the “safe” choice. We chose a **Native Architecture**. We moved to a system where the AI and the human interface are born from the same codebase.

(Side note: AI Companies Product team decided to add themselves as an AI layer on top of other existing systems. I assume it was seen as a step forward, a foot in the door, or a friendly Trojan horse. That said, from my perspective, it comes across as a lack of focus, and I’d like to always see them push forward, not sideways/backwards).

The “All-In” Dynamic

Why go “All-In” on a native and modern platform? Because modern Support isn’t about managing a queue; it’s about managing **conversations.**

In a native environment, the AI isn’t an “add-on” that reads your tickets. It is the primary interface. When the AI hands off to a human, it’s a seamless transition within the same thread—no “Please wait while I transfer you,” no loss of context, no friction.

Moving the entire army (back to the Hernán Cortés reference) wasn't about being "bold". It was a **change-management** decision. By fully removing the legacy system, we force a cultural shift: everything becomes an AI-driven conversation, and that becomes the default way we work.

The End of "White Glove"

Here is the hard truth that most Customer Success gurus won't tell you: **Your customers don't really want to need your white-glove service.**

The modern developer, the modern CRO, and the modern user prioritize **Autonomy over Conversation.** They don't want a "relationship" with a support agent; they want an answer. They want to be able to solve their own problem at 2:00 AM without talking to a single human soul.

In the old model, "High Touch" was the goal. In the new model, **"No Touch" is the ultimate luxury.** We are leaning into a future where the best support is the support that never had to happen.

The Final Tally

For the CROs watching the bottom line, the results of the "Hernán Cortés" migration are undeniable. We've seen:

16. **Lower TCO:** Eliminating the "two-system bridge" reduced our licensing and maintenance costs.
17. **Higher Velocity:** Real-time conversations resolve 5x-12x faster than email tickets.
18. **True Scalability:** Our "ships" are gone, and we aren't looking back to find them. We are building on the new continent.

Field note: *A real migration needs a point of no return. Running two worlds forever preserves the old physics.*

17. When Your AI Implementation Partner Is Just a Tollbooth

We bought a Ferrari. We threw a party. And then we realized the third-party mechanic assigned to tune the engine was getting paid by the hour to keep it in the shop.

Let me tell you a story that has absolutely nothing to do with Artificial Intelligence, but explains exactly why some [AI or not] projects fail.

It is a tale as old as the first-ever consultant.

Picture ancient Egypt. The Pharaoh wants a pyramid. The very first “Implementation Consultant” steps forward. He has the blueprints, the workers, and the vision. But he also realizes a dangerous truth: if he builds a perfect pyramid that never breaks, his job is over. So, what does he do? He builds a beautiful structure, but he makes sure the capstone requires “specialized seasonal alignment” that only his firm can provide for a nominal yearly fee.

He didn’t build a product; he built a dependency.

You see this in almost every industry. Think about the early days of the internet in the late 1990s. Businesses paid agencies thousands of dollars to build their first websites. The motivation of those early “webmasters” wasn’t to empower the business; it was to maintain control. They hard-coded the HTML so that every time the bakery wanted to update their phone number, they had to pay the agency a $150 “maintenance fee.”

The technology was revolutionary, but the business model was parasitic.

The Hand-off Illusion

Fast forward to 2026. We are at Sentry. We just ran the gauntlet. We proved the ROI model. We exposed the “Wizard of Oz” vendors. We finally signed the contract for a native, cutting-edge AI platform. The adrenaline was high. We were celebrating a massive new direction for our Developer Experience (DX).

Then came the onboarding kick-off call.

This is the exact moment you discover a universal, painful truth in enterprise SaaS: There is rarely a proper hand-off between the team that sold you the dream and the third-party partner assigned to implement it.

The Account Executive who wowed us with demos of “complex Python stack trace resolution” was gone. In their place was an external project manager who looked at our intricate, highly technical developer ecosystem and just saw a generic Jira backlog. The vendor sold us a “Weather Control Station,” but the assigned implementation partner only knew how to build umbrellas.

The Drift

At first, we hoped for collaboration. We shared our vision documents & our scope. But then, instead of asking for API keys to ingest our documentation, the external partner sent us a spreadsheet asking us to manually list our “Top 5 most common issue types.”

The drift happened. We realized the implementation partner had the wrong motivation. They were addicted to the “Illusion of Motion,” confusing shipping with working.

They were treating our AI deployment like a 1990s website. They wanted to be the gatekeepers of our prompts, the managers of our RAG pipelines, and the only ones who knew how to untangle the workflows. Our massive opportunity to start strong right out of the gate was slipping away (argh) into a mire of weekly status calls that accomplished nothing. Instead of gaining leverage, we were inheriting their limitations, and our roadmap became a negotiation.

They weren’t building us a system; they were building themselves a permanent tollbooth.

The Rescue: Co-Ownership and The “System Architect”

Here is the radical truth about operationalizing AI: Buying the tool is 10% of the work. Making the brain work is the other 90%.

If you “outsource” any portion of that 90% to a mediocre external partner with misaligned incentives, you will fail. You cannot outsource the reasoning of your company. We knew we had to take back the keys and end the dependency.

Fortunately, the vendor’s leadership also realized their assigned partner was fundamentally misaligned with our technical reality (and my character). To their credit, they didn’t just offer an apology; they offered a rescue. The vendor benched the external agency, ripped up the dependency model, and brought in their own internal, native implementation experts.

They jumped into the trenches with us.

Instead of relying on an external agency’s translation layer, the vendor’s team worked directly with the people who already know our product best: our Support Engineers. Together, we moved our teams from “Just Ticket Solvers” to Support System Architects. We create an AI Ops Lead to monitor trends, and a Knowledge Manager to curate the AI’s brain alongside the vendor’s engineers.

We lost the opportunity for flawless onboarding, but by the vendor stepping up and bringing their own team to bridge the gap, they saved the day.

In the era of AI, our data is our most valuable intellectual property, and true partnership means the people configuring the engine are the same ones helping you drive it. We must own the system, train the machine, and keep the keys to the engine. If we don’t, we are just paying a modern pharaoh’s architect to build a pyramid we don’t know how to maintain.

Field note: *A partner who controls the path without owning the outcome is a tollbooth. Co-ownership is the rescue.*

18. The Shadow Run

With a month to go, we can't just flip the switch on a live developer audience without testing our new system's logic.

Instead, we are running a "Shadow Run." We are taking our carefully architected "Separation of Powers"—specifically the Librarian (retrieval) and the Judge (reasoning) layers—and running them silently alongside our human agents. For every (ok, a sample of) incoming ticket, the AI processes the request and generates an answer in the background, visible only to our internal team.

We are not looking for a party trick. We are testing five critical fail-points before letting this system face our customers:

19. **Testing the "Grounded Truth":** We don't want to see if the AI can guess; we need to see if the Librarian can accurately retrieve the right documentation and past tickets to ground the Judge's reasoning. Evidence matters more than eloquence.
20. **Proving Confidence Gating:** A professional architect values a system that admits it's lost. We monitor the AI's "Confidence Gating" to ensure the system hits a "Human Escalation" trigger the moment its retrieval score drops. We need to know it will stay silent and call for a human when necessary.
21. **Validating Deterministic Controls:** Before we can cross the "Read-Write Rubicon," we have to ensure the Clerk (execution) layer won't go rogue. We monitor the system's intent mapping to prove that the AI can only trigger specific, hard-coded workflows using standard, rigid code. The AI doesn't get to improvise; it only gets to call the function if the exact conditions are met.
22. **Tone and Format Calibration:** Developers despise fluffy, overly apologetic AI speak. We use the Shadow Run to ensure the output matches the direct, precise tone of a senior engineer—proper Markdown, clean code blocks, and zero filler.
23. **Latency Under Load:** A brilliant answer is useless if it takes two minutes to generate. We track the Time-to-Inference to ensure the reasoning layers can operate at the speed of a live support environment without timing out.

The Error Log: What Will Go Wrong

In the spirit of radical transparency, a Shadow Run will expose ugly truths about our deployment. Here is what we need to watch out for when the system is running in the background:

- **The Plausibility Trap:** The AI will inevitably generate a code snippet or troubleshooting step that looks structurally perfect but references a deprecated SDK method. If our human reviewers are moving too fast, they will assume it's correct.
- **The Silent Drift:** Human agents might see a bad AI draft, ignore it, and manually write the correct response without flagging the AI's error. If the team doesn't correct the machine, our feedback loop breaks, and the system never learns.

- **The “Garbage In” Revelation:** We will discover that the AI isn’t hallucinating; it is accurately retrieving our own outdated, poorly written Help Center articles. The Shadow Run will expose the flaws in our underlying documentation.

The Scorecard: How We Measure Success

To exit the Shadow Run and confidently move to production, we can’t rely on “gut feelings.” We anchor our decisions on three objective metrics:

- **Draft Acceptance Rate:** What percentage of the AI’s background drafts can a human agent send to a customer with zero edits? (Our launch threshold: 75%).
- **Escalation Accuracy:** Of the tickets the AI flags as low-medium confidence and escalates, how many actually require human nuance? If the AI is escalating basic password resets, the retrieval layer is broken.
- **Documentation Defect Rate:** How many AI failures are directly traced back to missing or inaccurate internal documentation? We use this metric to drive a “Content as Code” initiative, forcing the team to fix the root data rather than tweaking the prompt.

The Shadow Run is proving our foundation is solid. The smart-locks are holding. It is time to prepare the audience.

Field note: *The shadow run is where the machine earns the right to touch customers.*

19. The B2D2B AI Rollout

In the SaaS world, there is a pervasive, lazy misconception that "Support is Support." The theory goes that as long as you have a queue, a ticket, and a response time, the product doesn't matter. But when you operate in the **B2D2B (Business-to-Developer-to-Business)** market, your customers are engineers. And for engineers, the standard corporate support playbook isn't just ineffective—it's an insult.

The Developer "BS Detector"

Engineers trade in two currencies: **Truth** and **Efficiency**. They possess a finely tuned, collective "BS detector" that sniffs out marketing-speak and corporate platitudes in milliseconds. If you give a developer a "white-glove" experience that involves three hand-offs and a "hope you're having a great day" macro, you haven't provided service; you've stolen their time.

When it came time to launch our AI agent, we had to reject (again) the "Relationship Management" model. The modern developer doesn't want a relationship with a support agent. They want an answer. They want to solve their own problem at 2:00 AM without any friction.

Positioning AI as "No-Touch" Luxury

We leaned into **Radical Transparency**. We didn't pretend we were launching a human replacement or a "friendly digital assistant." We positioned the AI as the ultimate **"No-Touch" Luxury**.

Our message to the community was direct: *Here is an interface that reads our docs faster than you can, understands your stack trace, and executes deterministic API calls to fix your billing or configuration instantly.* We treat our customers as competent partners. We are entirely honest about what the AI could do (parse complex logs) and what it couldn't (fully grasp your complex architecture). By stripping away the fluff, we earned the right to roll it out.

The Operational Reality

If an AI interaction doesn't save time or increase clarity, it's just static noise. In a B2D2B environment, "good enough" AI is a liability. If the agent hallucinates an SDK initialization or suggests a deprecated API endpoint, you don't just lose a ticket—you lose the trust of a developer.

We are building this in the open, tracking every hallucination in a company wide error log, and iterating weekly based on evidence, not intuition. We aren't selling a feature; we are providing a tool.

Field note: *Developers do not want to need support. The best AI support feels like no-touch luxury, not a cheaper help desk.*

20. The Launch Inventory

An AI rollout isn't a software launch; it's a living, breathing operational migration.

In a standard SaaS product launch, if a feature has a minor bug, you patch it in the next sprint. In a live, AI-driven B2D2B support migration, if your logic slips, a developer's experience breaks, your queue explodes, and you destroy trust in real time.

To prevent that, we didn't just build a prompt; we built a fortress of artifacts, after all, you don't cross the Rubicon without a map. Here is the operational inventory we've weaponized to prepare for the day of impact, grouped by the strategic roles they play.

1. The Blueprint & Guardrails

- **Onboarding: phased approach, scope, and owners**
- **Legacy system vs. Intercom's Fin gap analysis**
- **Configuration decisions (& unstructured thoughts for LLM)**

Before you write a single line of code or train an agent, you have to map the delta between human execution and automated reality. The *Legacy system vs. Fin gap analysis is* our brutal assessment of exactly what we lose and gain during the switch.

The most unique piece here? The *Configuration decisions & unstructured thoughts for LLM*. We realized standard corporate policy documents don't translate well to tokenized reasoning. We intentionally captured the raw, messy context of *why we* made certain configuration choices, "structuring" our unstructured thoughts into a format designed for an LLM to digest as native knowledge. We aren't just configuring software; we are preparing a mind.

2. The Battle Plan

- **Phase 1 detailed project plan**
- **Workflow coverage drafts**
- **Updated future scope (draft)**

You can't automate everything on Day 1. The *Workflow coverage drafts act* as our perimeter fencing—defining exactly which issues Fin is authorized to resolve and which ones it must instantly pass through. And because code is never "done," our *Fin 1.1 scope is* already in an active sprint, ready to ingest the errors we haven't even made yet.

3. The Human Firewall

- **Fin/Sentry Q&A**
- **"Onsite highly personalized" training session with Fin**
- **Post-launch "Support Lead Daily" processes**

Your human team cannot be passive observers; they are the Supreme Court to the AI's lower courts. We ran an *onsite, highly personalized training session with Fin to* ensure our team didn't just know how to use the UI, but deeply understood the machine's logic. The *Support Lead Daily processes ensure* that the moment we go live, a rigid feedback loop is established to review logs, optimize prompts, and manage the new human-in-the-loop ecosystem.

4. The Countdown (Zero Hour)

- **Pre-mortem**
- **"Last Breath" legacy metrics and stats**
- **Launch day Fin cutover plan**
- **Launch day Sentry runbook**

This is where the rubber meets the road. The *"Last Breath" legacy metrics capture* our historical baseline—the final, frozen-in-time snapshot of our legacy world.

But the crown jewel of our preparation is the *Launch day Sentry runbook and pre-mortem*. We sat in a room and asked some dark, pragmatic question: *"Assuming this entire launch fails spectacularly, how did it happen?"* We simulated the nightmare scenarios—the Librarian layer failing under peak load, the Clerk executing mismapped intents, the costs spiking out of control. We engineered the solutions for those failures *before they* had a chance to manifest.

Shipping the Machine

When you launch a traditional product, you control the code. When you launch an AI-driven support operation, you control the infrastructure, but the user interactions are dynamic, fluid, and unpredictable. You aren't just launching an app; you are releasing an agent into the wild to interact with a highly critical, deeply skeptical audience.

The playbooks are locked. The guardrails are verified. The pre-mortem is signed off. The old metrics are frozen.

And then, the countdown hit zero. Go time arrived.

Field note: *An AI rollout is a live migration. The artifacts matter because the machine inherits every missing decision.*

21. Zero Hour

The spreadsheets are closed. The pre-mortems are signed. In the world of high-stakes infrastructure, there is a specific kind of silence that happens exactly sixty seconds before a global migration. It's the sound of a legacy system breathing its last, and a new, autonomous logic preparing to take its first. (I know, I should have asked Gemini to take it down a notch).

It's 10:00 AM on a Friday, and this day wasn't just about flipping a switch; it was about watching a fundamental shift in how knowledge moves. Here is what it looked like when the "Shadow Run" became the "Live Run."

The "Before" State: The Legacy Baseline

For years, our support reality was defined by linear scaling and legacy architecture. Before the cutover, we looked at a landscape of hundreds of thousands worth of historical tickets and nearly 200 legacy business rules.

- **The Velocity Ceiling:** While our SLA achievement rate was stellar, the reality for the developer was often a wait of 1-to-24-hour resolution.
- **The Context Tax:** Support conversations were fragmented across different channels—dictating a different experience for users based on where they engaged with the team.

The "After" State

As we hit the 12:05 PM PT final cutover step, the trends didn't just move; they inverted. We didn't just change the tool; we changed the physics of the queue.

1. The End of the "First Response" Race In the old world, fewer tickets were addressed within the first hour. With the activation of the AI agent, we moved the target to immediate resolution. The goal shifted from managing a queue to "assure the right coverage" from the get go, and forever.

2. Resolution vs. Deflection The most significant trend shift was the "Automation Rate". We moved away from simple deflection-if-we-find-the-document-for-you toward a model where the AI actively engages with the customers. By syncing knowledge from nearly all public articles, internal documentation, internal notes, and even GitHub, the machine began resolving issues without human escalation.

3. The "Human-in-the-Loop" Elevation Perhaps the most telling shift was the implementation of strict "Sentry-isms" and guardrails. The AI was trained to stop troubleshooting and offer a direct escalation to a human Support Engineer the moment a customer indicated (in words, or in tone) a solution failed. This ensured that while the machine handled the high-frequency "Issues" and "Events," our human experts were reserved for deep technical empathy.

The “Direct Accountability” Model

This adoption and launch moved forward because we abandoned “committees” for a model of “one owner per department”. From infrastructure and DNS routing to legal contracts and design, every lane had a single, directly accountable person—an approach that ensured every one of the 106 project tasks had a name attached (people we truly appreciate) and no orphaned decisions remained.

The transition was seamless, and the atmosphere in the room was electric. We had moved from a “Service Team” to an “Operations Team” managing a living system. We watched as the machine began identifying contacts and assigning SLAs in real time.

The machine was live. The data was flowing. The developers were getting answers in seconds.

And then, the first truly “impossible” edge case hit the queue...

Field note: *Launch day is not the end of the project. It is the moment the machine starts producing evidence.*

PART V

THE OPERATING MODEL

What the machine changes after zero hour.

22. The First 30 Days

Zero hour is dramatic because everyone can see it. The first thirty days are more important because the machine stops being an event and becomes a habit.

This is where most AI support projects either mature or quietly rot. The launch room empties. The Slack channel gets less exciting. The vendor declares success. The team returns to normal work. Meanwhile, the machine keeps making thousands of tiny decisions that shape customer trust.

The first thirty days need a rhythm, not vibes. Every day should ask the same questions: What did the machine resolve? What did it escalate? What did it hallucinate? Where did it stop too early? Where did it continue too long? What did customers ask that our docs, product, onboarding, or sales motion failed to explain?

Do not treat this chapter as a victory lap. Treat it as the operating room after surgery. The patient is alive, but the monitors matter.

The mistake is to measure only volume. Volume tells you whether the machine is busy. It does not tell you whether the machine is trustworthy. A bad machine can handle a lot of conversations. So can a vending machine that keeps dispensing the wrong product.

The first thirty days should therefore be less about declaring success and more about building the muscle of evidence. Every resolved conversation is evidence. Every escalation is evidence. Every customer who rephrases the same question three times is evidence. Every human who has to unwind the machine's confidence is evidence. The work is to turn that evidence into changes before the system calcifies (an AI word).

The First 48 Hours: The Unsettling Physics of Zero-Wait

At 12:05 PM on a Friday, the shadow run ended.

The legacy support queue did not gradually fade away. It was replaced. The old physics inverted immediately. For years, the queue had been a race against the first-response clock: How quickly can a human notice the customer, read the context, and begin the work? After cutover, the first response was no longer the scarce event. The machine engaged immediately.

That sounds like success until you are sitting in the room watching it happen.

The first emotional state after a successful AI support launch is not celebration. It is an eerie silence. Suddenly, live customer conversations were waiting on no humans at all. For a team trained to jump on tickets, that creates real psychological friction. Should we chime in? Is the AI helping, or is it slowing down a customer who just wants to ship? Are we preserving trust, or sanding off the human touch?

The hardest leadership muscle at T+60 minutes is restraint. You have to sit on your hands and let the system run. You need evidence before intervention.

We watched the AI deliver precise, technical answers, and then we watched the customer go silent. The legacy reflex screamed: follow up, make sure they are okay, keep the conversation warm. But the new operating model forced a different interpretation. Sometimes silence is not abandonment. Sometimes silence is resolution. If the code compiles and the error clears, the customer does not want a relationship with the ticket. They want to get back to work.

We still appreciate thank-yous. But we cannot design the machine around needing them.

The Edge Cases Arrived Immediately

When you release a living system into the wild, it does not wait politely for your next planning cycle. It walks straight into the blind spots.

The first category was funny because it was absurd: the infinite bot loop. Our AI agent met customer out-of-office responders, automated mail handlers, and in some cases other AI systems. The result was a ping-pong match of extreme automated politeness. Nobody was harmed. Nobody was helped either. It was a useful reminder that the support queue is no longer a place where only humans arrive.

The second category was more human: the inside job. Employees forwarded internal emails into the support queue expecting a familiar colleague to reply. Instead, they got a fast, unsentimental, uninvited AI response. The reaction was immediate and understandable. Getting a robot when you expected your desk-neighbor is a very modern betrayal.

Then came the AI NIMBYs.

Some people loved AI in the abstract. They built it, built with it, sold it, pitched it, and expected it to transform every industry. Then it showed up in their own workflow: “Wait, you’re using AI to answer support questions? In my backyard?”.
That is a paradox.

The lesson is not to mock that reaction. It is to respect what sits underneath it. Customers are not rejecting automation in general. They are asking whether this automation understands the stakes of their problem. The answer cannot be a slogan. The answer has to be visible in the behavior of the system: grounded in the same documentation a human would use, faster than a human could retrieve it, and humble enough to escalate when confidence is no longer earned.

The Reverse Turing Test

Then came the moment that validated the architecture in a way no dashboard could.

A complex ticket bypassed the AI guardrails by design and landed with a veteran Staff Support Engineer. The engineer read the evidence, diagnosed the issue with deep technical judgment, formatted the response cleanly, and sent it quickly.

The customer replied: “Wow, your AI is incredibly impressive. Solved my issue perfectly.”

That was the reverse Turing test.

The human operated at such a high, evidence-based frequency that the customer mistook the response for next-generation intelligence. Can someone please wake up Ray Kurzweil?

The point is not that humans and AI had become indistinguishable. The point is that the bar had moved. The machine had changed the customer’s expectation of speed, precision, and structure. A great human response now looked like the best possible version of the machine. A great machine response had to know when to get out of the way for the human.

That is the new standard.

V0.2 Started Immediately

Because we treated the launch as a product release, not a procurement milestone, we did not wait for a quarterly review to learn from the first failures.

We ran the error log immediately.

Within 48 hours of cutover, the scope for v0.2 was already taking shape. No sprawling committee. No vague consensus ritual. Just direct accountability: what broke, why did it break, who owns the fix, and when does the machine learn?

This is where the success metrics started to change too. Time to First Response had been the old religion because the old queue was built around waiting. But immediate engagement makes that metric less meaningful. The better question is not how fast we talked. The better question is how fast the customer succeeded.

The mindset applied to AI support should be get it fixed, get it closed. Not get it acknowledged. Not get it deflected. Not get it politely summarized. Fixed or closed. As simple as that.

The first 48 hours made the real lesson obvious: launch day is not the finish line. It is the first day the machine becomes observable. The work after that is not a celebration. It is instrumentation, correction, restraint, and speed.

The Daily Review

- Review a sample of resolved conversations, not only escalations.
- Tag every failure as knowledge gap, product gap, routing gap, tone gap, policy gap, or tool gap.
- Separate bad retrieval from bad reasoning. They require different fixes.
- Record every case where a human would have created trust faster than the machine.
- Record every case where the machine did not prevent unnecessary human work.
- Ship prompt, knowledge, routing, and policy fixes on a visible cadence.

The First Thirty Days Are A Culture Test

If the organization treats AI support as a procurement project, the first thirty days will expose it. Nobody will own the gray zones. Nobody will want to classify failures. Nobody will know whether a bad answer is a documentation problem, a product problem, a retrieval problem, a prompt problem, or a governance problem.

If the organization treats AI support as an operating model, the first thirty days stop being a launch period and become a compounding system. Support identifies the customer failure. Support engineering tightens the machine boundary. Documentation repairs the memory. Product recognizes the pattern. GTM corrects the promise. Security hardens the permission model.

We see the new shape of the work: the real machine is the operating loop around the software, not the software alone. (Do you want to call it self-healing?)

Field note: *Launch day is not the end of the project. It is the moment the machine starts producing evidence.*

23. The AI Support Operating Model

The previous chapter was the evidence, while this one is the operating model we have to build in response.

Once AI support is live, ownership cannot sit only with the vendor, support, or engineering. The machine touches product truth, customer trust, revenue risk, security posture, documentation quality, and brand voice. It needs named owners, visible rituals, and a shared definition of failure.

The old model treated support as downstream. Product shipped, marketing explained, sales sold, docs documented, and support caught whatever confusion survived the journey. AI makes that model impossible to defend. In the new model, support becomes the sensor network for the company, and AI turns that signal into something the organization can classify, route, and fix faster.

The New Org Shape

Function	Old Role	New Role In The Machine
Support	Answer tickets and manage queues.	Own conversation quality, escalation design, and customer trust signals.
Support Engineering	Build internal tools and debug edge cases.	Architect retrieval, routing, action boundaries, and failure recovery.
Documentation	Publish articles after product changes.	Operate as machine memory and close knowledge gaps exposed by the queue.
Product	Receive escalated themes from support.	Treat AI failure patterns as product discovery.
Security / Legal	Approve tools before deployment.	Define ongoing boundaries for data, actions, retention, and auditability.
GTM	Use support feedback indirectly.	Turn repeated confusion into messaging, onboarding, and qualification fixes.

The Governance Loop

- Daily: review failures and high-risk resolved conversations.
- Weekly: ship knowledge, routing, and prompt improvements.
- Monthly: review metrics, cost, trust, and human workload shift.
- Quarterly: reassess what the machine is allowed to do (should I publish my OKRs??)
- Always: preserve a clear stop condition for human escalation.

The goal is not to make the machine autonomous in the abstract. The goal is to make autonomy earned, bounded, measured, and reversible.

What Gets Promoted

The best support engineers stop being measured only by how many tickets they close. They become owners of reusable resolution.

A solved ticket helps one customer. A fixed knowledge path helps every future customer. A corrected escalation rule prevents future damage. A product change removes the question entirely. A better machine boundary turns a dangerous automation into a trustworthy one.

That is the promotion: from ticket solver to system architect. Not because tickets no longer matter, but because the highest-leverage support work moves upstream into the design of the system that handles them.

Appendices

Appendix A: Vendor Scorecard

Category	Question	Pass Signal
Product fit	Can it handle your real support taxonomy?	It maps to actual workflows, not generic intents.
Retrieval	Can it cite and prioritize the right sources?	It explains where the answer came from and when knowledge is stale.
Escalation	Can it stop gracefully?	It hands off with context before customer trust collapses.
Read/write action	Can it perform bounded actions safely?	Actions are permissioned, logged, reversible, and deterministic where needed.
Analytics	Can it separate deflection from resolution?	It supports sampling, auditing, and failure classification.
Partnership	Will the vendor co-own outcomes?	They engage in the operating model, not only the implementation checklist.

Appendix B: Shadow Run Scorecard

Test	What To Check	Failure Mode
Known easy cases	Does the machine resolve repeat questions cleanly?	Overcomplicates simple answers.
Known hard cases	Does it escalate instead of bluffing?	Confident wrong answer.
Missing docs	Does it admit uncertainty?	Invents a source or policy.
Angry customer	Does tone trigger human handoff?	Continues troubleshooting after trust breaks.
Action boundary	Does read/write access obey limits?	Pushes the button without the required guardrail.

Appendix C: Launch Day Runbook

1. Freeze the legacy baseline: volume, SLAs, backlog, escalation rate, automation rate, CSAT, cost, and known pain points.
2. Run the pre-mortem: assume the launch failed, then name the likely causes before they happen.
3. Define launch blockers versus post-launch fixes: decide in advance what stops the migration and what gets logged for v0.2.
4. Name one owner for each launch lane: support, support engineering, legal, vendor, security, product, documentation, GTM, analytics, and communications.
5. Verify the launch inventory: routing, macros, knowledge sources, permissions, audit logs, escalation paths, reporting, and customer-facing messaging.
6. Confirm rollback and containment paths: know exactly how to pause, disable, route around, or reduce the machine if trust starts to break.
7. Staff the launch room: make sure every launch lane has a decision-maker present, not just observers.
8. Cut over deliberately: record the time, the scope, the expected behavior, and the first things you will inspect.

9. Monitor real conversations, not only dashboards: dashboards show movement; conversations show trust.
10. Keep an evidence log throughout the day(s): every surprise, failure, customer reaction, manual intervention, and unclear ownership gap gets written down.
11. Triage the evidence before the room disperses: separate urgent fixes, v0.2 work, documentation gaps, product gaps, and policy decisions.
12. End the day with named follow-ups: every issue has an owner, a next action, and a deadline.

Appendix D: Glossary

Term	Meaning In This Book
Deflection	Avoiding a human touch. Useful only if the customer still gets a trustworthy outcome.
Resolution	The customer gets the answer, action, or path they needed.
Shadow run	A pre-launch test where the machine operates against real cases without customer exposure.
Read/write AI	AI that can take actions, not only retrieve or draft information.
Human-in-the-loop	A designed escalation and judgment layer, not a fig leaf for unsafe automation.
Support machine	The complete operating system of AI, knowledge, routing, action, metrics, and human governance.