

Why don't we just build it ourselves?



Should you still buy enterprise software when you can build it yourself?

A founder's perspective.

Patrick O'Connor

Founder and CEO, Gatekeeper



No More Surprises.

www.gatekeeperhq.com

Why don't we just build it ourselves?

It's the question every software vendor dreads.

Inside, an honest answer from Gatekeeper's CEO, Patrick O'Connor.

01 Foreword

Why this question deserves a straight answer

02 The pressure to build

Boardroom pressure, vendor-scale hype and SaaS fatigue

03 The reality of building

The code, the lifecycle and the five percent you cannot see

04 The middle ground

Build what makes you different, buy what makes you run

05 Appendix: fourteen questions before you build

Score each honestly before committing engineering effort

06 One unified solution

Vendor onboarding and contracting in one workflow, on one unified record



Patrick O'Connor,

Founder and CEO, Gatekeeper

September 2026.

01 Foreword

I have spent years in this market, first as a consultant inside some of the world's largest organisations, and now as the founder of the software company Gatekeeper. I have watched companies build brilliantly, and I have watched them build themselves into a corner. I've found the difference between the two isn't talent, but knowing which question to ask before you start.

Finance might ask, why are we renewing software? Procurement will wonder what they'll inherit next, but every software CEO is beginning to be asked a question they'd rather avoid during the sign-off process. It's usually late, and usually from someone in IT or transformation: why don't we just build it ourselves?

Most vendors will dodge it, change the subject, or talk about total cost of ownership and hope the thought behind the question passes.

I want to do the opposite. It's a serious question and deserves consideration. For the first time in enterprise software history, building your own systems is genuinely possible for a normal company. It's changing the conversation, and to pretend otherwise insults everyone's intelligence.

This guide gives you an honest answer from a founder at a software company. Where building makes sense, I will tell you. Where it's a trap, I will tell you that too, and why. And where a path exists that most of the market hasn't noticed yet, I will show you what it looks like.



02 The pressure to build

Every CFO I speak to is under the same pressure. Somewhere above them, or beside them, someone is banging a fist on the boardroom table, asking where the AI savings are.

That pressure has to go somewhere. And right now, it's flowing into one seductive idea: stop buying software, start building it.

The logic feels irresistible. Your organisation has bought an enterprise licence for a frontier AI model. The spend is significant, the token bills keep arriving, and nobody can point to a return. So the people managing the purse strings do what they have always done. They look for costs to cut and value to claim. SaaS subscriptions are the obvious target, and building AI replacements is the obvious justification.

Look closer at where that idea came from, though, and it starts to wobble.

The AI companies themselves need you to believe it. To justify valuations measured in the hundreds of billions, their addressable market has to be effectively all software, all work, everything. So the rhetoric from the top of the industry has been relentless: the age of applications is over, everything will be generated on demand; why would anyone pay for software again? It's a story told by people selling tokens, and so I would ask you to consider the source first and foremost.

Then there is the demo effect, where, for the first time ever, a non-engineer can describe an application out loud and watch it appear. It's genuinely exciting. I have felt it myself. You watch an interface assemble itself in minutes and think, hang on, what am I paying vendors for?

And underneath both of those sits something the software industry has to own. We have not always delivered great value. Companies bought a lot of software over the last decade, and plenty of it disappointed. In your personal life, when a subscription disappoints, you cancel it that afternoon. In B2B, you're locked in for two and a half years at numbers with a lot of zeros. That resentment is real, and it fuels the cycle.

Put those three together, boardroom pressure, vendor-scale hype and accumulated SaaS fatigue, and you get the current climate: a widespread belief that building is the fiscally responsible choice and buying is the lazy one.

THREE FORCES BEHIND THE SAME CONCLUSION

Boardroom pressure

An enterprise AI licence is being paid for, and nobody can point to a return.

Vendor-scale hype

To justify the valuations, the addressable market has to be all software, all work, everything.

SaaS fatigue

A decade of software that disappointed, on contracts you can't cancel that afternoon.

CASE IN POINT • KLARNA

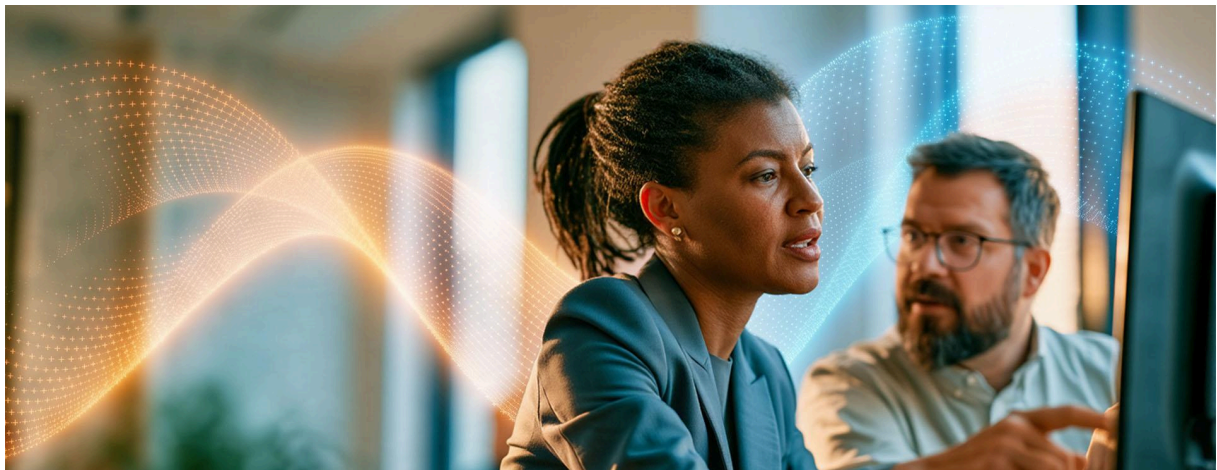
In 2024, [Seeking Alpha reported](#) that Klarna's CEO, Sebastian Siemiatkowski, announced that the company was shutting down SaaS providers Salesforce and Workday, and instead was pursuing a combination of AI, standardisation, and simplification, and the headlines wrote themselves. What followed drew far less coverage.

Sebastian Siemiatkowski clarified that "[we did not replace SaaS with an LLM](#)", noting that storing CRM data in an LLM "would have its limitations". CX Today reported that some of that work moved to different SaaS vendors, stating Klarna "[had pulled together other SaaS tools to replace its CRM functionality](#)," and he said he doubted others would follow.

On customer service, Klarna revealed that an AI assistant was doing [the work of around 700 service employees](#), before it went back to recruiting human staff, with [Siemiatkowski explaining](#) that cost had been "a too predominant evaluation factor when organising this, what you end up having is lower quality." Klarna was one of the most-quoted examples of building your way out of software, and it is one its own author has qualified.

A pattern exists here, and it's as old as enterprise software. The pressure to build is usually strongest exactly where the case for building is weakest: in the plumbing, the operational systems, the things every company needs and no company differentiates itself on.

The pressure is real, but the logic is not.



03 The reality of building

The demo is not the product, and that's the whole argument, but let me earn it here and now.

When you watch an application assemble itself from a voice prompt, you're seeing an interface and a database. Those are the two easiest parts of enterprise software, and they are perhaps, in my opinion, only 5 percent of it. Everything that makes software safe to run a business on is invisible in the demo.

That looks like audit trails. User provisioning and deprovisioning. Enterprise backup and recovery. Data sovereignty, and knowing which jurisdiction processes your data. It's disaster recovery that has actually been tested and independent accreditation, which, at Gatekeeper, means ISO 27001, ISO 9001, SOC 1 Type 2 and SOC 2 Type 2, audited annually by people whose job is to catch us out. None of that shows up when you vibe code a contract repository on a Friday afternoon. All of it is what an auditor, a regulator or an acquirer will ask for first.

WHAT THE DEMO SHOWS, AND WHAT IT DOES NOT

Visible in the demo: the interface and the database	5%
Invisible in the demo, and the first thing an auditor, a regulator or an acquirer asks for	95%

Audit trails	User provisioning and deprovisioning	Enterprise backup and recovery
Data sovereignty	Disaster recovery that has been tested	Independent accreditation

The code itself.

AI-generated code works, in the sense that the interface responds and the buttons do things. But really good code is concise. It's the engineering equivalent of the old line: if I had more time, I would have written you a shorter letter. AI code is the opposite: verbose and sprawling, and it carries a specific trap.

When no human wrote it, and no human understands it, no human can fault-find it, which inevitably means you're debugging your own core systems by asking a model to guess.

The lifecycle.

Software is not an event but a rhythm: ideation, planning, design, estimation, development, QA, staging, production, maintenance, evolution. Companies like Gatekeeper run that rhythm across hundreds of customers on a single codebase, which is the model's entire economy, and the cost of every improvement is spread across everyone.

THE RHYTHM YOU TAKE ON

Ideation	Planning	Design	Estimation	Development
QA	Staging	Production	Maintenance	Evolution

THE ECONOMY OF ONE CODEBASE

BUILD · HOW YOU RUN IT

One codebase, one customer: you.
You carry the full cost of a one-off, forever.
The rhythm becomes your job, forever.

BUY · HOW IT RUNS

One codebase, hundreds of customers.
The cost of every improvement is spread across everyone.
The rhythm is someone else's full-time job.

Build your own, and you carry the full cost of a one-off, forever. Every internal app is a one-hit wonder, and someone has to keep touring it.

And someone is the operative word when the person who built it eventually leaves. The code lives on a laptop, or in a repository nobody else has touched. The data is loaded, the team depends on it, and now nobody can access it, extend it or fix it. I've seen this happen inside our own walls at Gatekeeper: a talented person builds an impressive internal platform as a side project, then moves on, and the first question is the one nobody asked at the start. Who looks after this now?

If you operate in a regulated market, all of this stops being an efficiency argument and becomes an existential one.

Picture the scenario. You're a month from a regulatory review or audit. You need to demonstrate your third-party population, the contracts governing those relationships, the risks across your supply chain, and the evidence that those risks are being appropriately managed. Let's say your home-built system is 95 percent accurate.

5%

The problem is that you don't know which 5 percent is wrong, or whether that 5 percent contains the relationship, obligation or risk a regulator is about to ask you to evidence.

The consequence is not theoretical either. In regulated markets, the missing few percent is a ruinous fine, and in the worst case it is your licence to trade.

There's a design principle buried in that which almost nobody building their own system thinks about: the weighting of errors. A system that wrongly rejects a vendor has caused an inconvenience. The vendor reapplies, someone reviews it, life goes on. A system that wrongly approves a vendor has caused a liability that may surface two years later in front of a regulator. You have to weight the whole system towards false rejection and away from false approval. You only know that asymmetry if you have spent years watching how these processes fail. As good as the AI is, you don't know what you don't know, and that's why expertise is the product.

NOT ALL ERRORS COST THE SAME

FALSE REJECTION

An inconvenience. The vendor reapplies, someone reviews it, life goes on.

FALSE APPROVAL

A liability that may surface two years later in front of a regulator.

One last piece of evidence, and it's the one I find most telling. Gatekeeper is a software company, and building software is literally the thing we do. And we've vibe-coded almost nothing of operational significance. We recently implemented NetSuite, and we could absolutely have written our own finance system, but we didn't. That's because every hour spent building non-core software is an hour taken from what our customers pay us for.

If a software company will not build its own non-core software, that should tell you something about whether an insurance company should.

04 The middle ground

On the middle ground, I'm breaking ranks with my own industry. The answer to build versus buy is not buy but to build and buy, and split it correctly.

The rule can be summed up in one sentence: build what makes you different, buy what makes you run.

David Friedberg tells the best version of this story against himself. Deep in AI, running an agricultural science business, he described on the [All In podcast](#) how his team spun up its own CRM over a weekend using AI coding tools. Then reality arrived, one feature request at a time. Accounts. Security. Access controls. A protected data repository. A year later, he had gone full circle, implemented Salesforce, and sat in Marc Benioff's office for the Salesforce earnings call to tell the story. His conclusion was not that building was wrong, but that he had built the wrong thing. As he put it, the question is where your best return on time and money sits, and for his company that is plant breeding technology, the thing that makes them unique, not recreating a CRM that already exists.

This is the whole framework anyone looking to build should work from. If you are a biotech, pour your build effort into how you manage patents and products. If you are an insurer, build around claims, underwriting, and your edge. Anything genuinely unique to how you win deserves your engineers, your tokens, and your attention, because software in that space amplifies your value proposition.

WHERE THE SPLIT FALLS

BUILD • HOW YOU WIN

Biotech: how you manage patents and products.
Insurer: claims, underwriting and your edge.
Agricultural science: plant breeding technology.
Anything genuinely unique to how you win.

BUY • HOW YOU RUN

Managing your contracts.
Onboarding your vendors.
Tracking your third-party risk.
The finance system, and the rest of the plumbing.

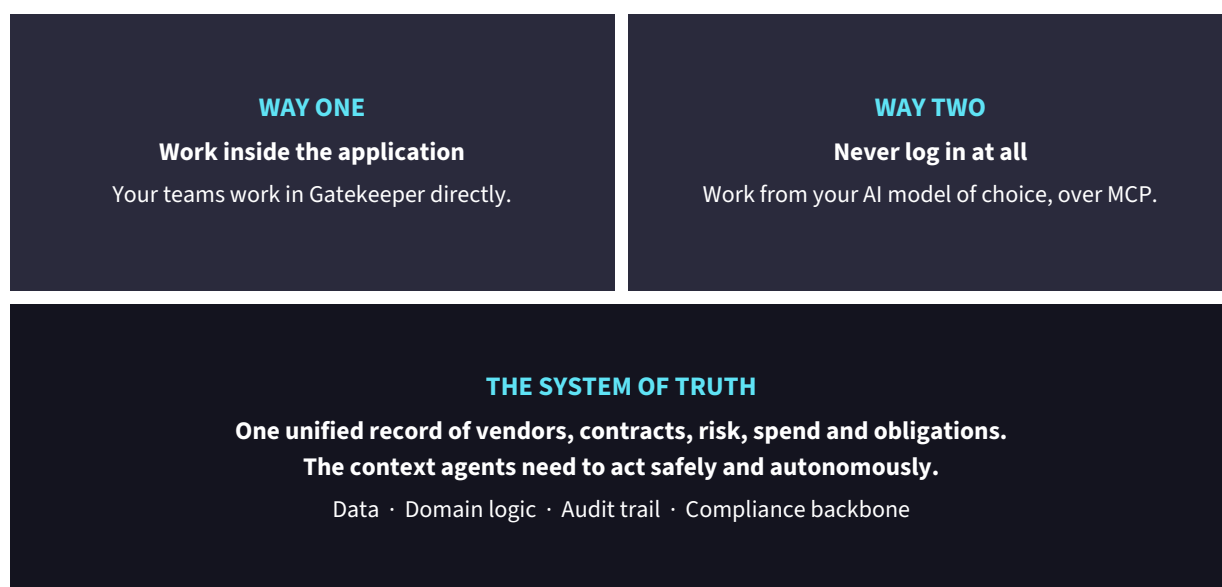
Every yes is a thousand nos.

In vendor and contract management? Managing your contracts, onboarding your vendors, tracking your third-party risk? That's not what you do; a unified vendor and contract lifecycle management solution does it, so you can do the work that makes you different. And every hour your best people spend building and maintaining plumbing is an hour taken away from your edge. Every yes is a thousand nos. Saying yes to building your own contract repository is saying no, a thousand times over, to the work only your company can do. You're either out of resources or out of ideas, and nobody is out of ideas. I could clean my own cars. I

don't, because the value of my working hour is higher than the cost of the wash, and if it weren't, I should be cleaning cars for a living.

You don't have to choose between your current AI investments and a solution like Gatekeeper, and it means you can work in two ways: work inside the application if you want to, or never log in at all: connect from whichever AI model your organisation has standardised on, through [open protocols like MCP](#), and use the platform's data, functionality and agents from there.

TWO WAYS IN, ONE SYSTEM OF TRUTH



Build versus buy used to be a binary because buying meant surrendering: your data locked in someone else's interface, your processes bent around someone else's workflow. That's no longer true, and this is where the middle ground gets interesting. That's the part most of the market has not caught up with. It now takes more than one form.

The first is buying differently.

This pattern is emerging across serious software, and it now has proof at market scale: Salesforce launched its version, Headless 360, earlier this year. Gatekeeper works the same way. It holds the data, domain logic, audit trail, and compliance backbone. It's one system of truth: not a passive repository but one unified record of your vendors, contracts, risk, spend and obligations that agents can safely act on.

So this dissolves the dilemma you're facing. Your central AI investment, the enterprise licence the CFO is asking hard questions about, gets a concrete, measurable use case: your teams driving vendor and contract work from the model they already use, against a foundation that is accredited, audited and maintained by someone whose entire business depends on it. You get the flexibility that made building attractive without inheriting a one-off codebase that dies when its author resigns.

And it answers the question nobody asks about AI efficiency gains:
where does the saved time go?

42%

BCG's study, [AI at Work: Strategy Matters More Than Tools](#) (June 2026), puts numbers on it: "Of frontline workers, the 42% who are regular AI users report saving a workday each week", and "more than two-thirds of all employees of all types say AI has taken over simpler tasks, leaving them with more complex work". However, "66% still receive limited or no guidance on what to do with the time they save".

A day a week only becomes money when you redirect it deliberately: into vendor consolidation, strategic sourcing, and renegotiating contracts that auto-renewed while everyone was firefighting.

The second form is building differently.

This is the orchestration layer. Not everything worth building is another system, and there is real value in a thin layer across the systems you already run, aimed at the gaps between them, where the work breaks. One question separates legitimate from dangerous.

When the orchestration layer acts, where is the record?

A home-built layer has no audit trail worth the name. Each system beneath it logs its own slice, and nobody logs the whole, so the most consequential logic in the stack, the decisions that span systems, lives in the least governed component.

In a regulated market, there's a harder stop. A fragmented audit trail is not an audit trail. When the regulator or auditor asks how a vendor was approved and what live contracts they have today, they want one complete record: who decided, on what evidence, in what order, and what that relationship now governs. That isn't a story reassembled from the logs of five systems.

The second question is the one that catches people, because the vendor sits in one system and the contracts in another, and the join between them is exactly what fragments can't produce. Fragments don't evidence a decision. They evidence that nobody actually owned it. So in regulated markets the home-built layer over fragments isn't really an option at all. The moment the layer holds state, stores records, or approves things, you haven't integrated your systems; you've built a new system of record without the accreditation.

The same test applies to the commercial version. A growing category of venture-backed tools now sells orchestration across your existing stack: intake, approvals and workflows routed over the top of the systems you already own. The routing is governed, but the record beneath stays fragmented. If you ask where the vendor and the contract actually live once the workflow completes, the answer is still five systems and a spreadsheet. A single pane of glass is a veneer. It lets you see everything but owns nothing, and the moment an auditor looks at it, the fragments are all still there. Looking unified is not the same as being unified.

In a regulated market, it's a finding waiting to happen. An orchestration layer inherits whatever sits underneath it. Governed, permissioned, auditable, unified substrate, and the layer compounds it.

Fragmented records and blind spots, and the layer industrialises the mess. Orchestration only becomes an answer when the workflow runs on one truly unified record.

Whichever form you take, neither building nor buying fixes the failure mode alone. The barrier to building has fallen so far that ungoverned builds now happen below the waterline: data pasted into a personal AI chat, an app deployed with hard-coded credentials, an automation wired to a spreadsheet, none of it signed off, none of it visible until something goes wrong. These workarounds are born from daily friction, and not from any decision a review board will ever see, so no purchasing choice prevents them and process only partly does. The practical defence is to make the sanctioned path the path of least resistance. When governed data is queryable from the tools people already use, the workaround loses its reason to exist.

So when someone in your sign-off process asks: why don't we just build it ourselves, here is my honest answer: You should build, aggressively and ambitiously, the things that make you unmistakably you.

And you should refuse to let a single engineering hour leak into rebuilding solved problems. Focus is the only resource you can't buy back.

Build your edge, buy your foundation, and connect the two.



05 Appendix:

Fourteen questions before you build

Score each honestly before committing engineering effort to an internal build. The more “no” answers, the stronger the case for buying.

STRATEGY

1. Is this system part of how we win customers, or part of how we run the business?
Build the first. Buy the second.
2. If this project succeeds completely, will a customer ever notice?
3. What would the same engineering hours produce if pointed at our actual product or service?

REALITY OF OWNERSHIP

4. Who maintains this in year two, and is that in their job description?
5. If the person who built it resigns tomorrow, can anyone else access, understand and fix the code?
6. Have we budgeted the ongoing cost: maintenance, upgrades, security patching, user support?
A build is a subscription you pay in salaries.

ENTERPRISE GRADE

7. Can this system produce a complete audit trail on demand?
8. Does it handle user provisioning, deprovisioning, backup, recovery and data sovereignty to the standard our auditors and customers require?
9. Could it pass the accreditations we ask of our own vendors (ISO 27001, SOC 2 or equivalent)? Would we approve ourselves in our own due diligence process?

RISK

10. If this system is wrong 5 percent of the time, what is the cost of the worst 5 percent?
In regulated processes, weight errors asymmetrically: a false rejection is an inconvenience, a false approval is a liability.
11. A month before an audit, can we produce every record this system holds, evidenced, without the person who built it?
12. What workarounds have people already built around this process, and will the option we choose remove the friction that created them or add to it?
The spreadsheet and the personal AI chat are telling you where the friction is.

THE MIDDLE GROUND

13. Before building, have we checked whether a specialist platform already offers headless or MCP access, so our own AI models can work directly against one accredited system of truth?
If yes, the build case usually collapses.
14. If you build an orchestration layer, what does it orchestrate across, and when it acts, where is the record?
If the answer is inside the layer, you have built an unaccredited system of record.

06 One unified solution

Gatekeeper runs vendor onboarding and contracting in one workflow, on one unified record. No other platform does both in one place, but what does that mean to you?

Separate systems are costing you. Each is good at its job, but the gap between them is where the problems start. Risk clears in one system, contracting in another, and waiting moves from days to weeks. Due diligence finds something, and the clause it should produce never gets written. Legal, finance, and compliance each answer from different perspectives, and so the answers never match.

WHERE THE GAP BETWEEN SYSTEMS SHOWS UP

**Risk clears in one system,
contracting in another.**

Waiting moves from days to weeks.

**Due diligence finds
something.**

The clause it should produce never gets written.

**Legal, finance and
compliance each answer
from different perspectives.**

The answers never match.

Gatekeeper is one system of truth across vendors, contracts, risk, spend and obligations. It means all that data lives in one record, where risk shapes the contract, spend runs against it, no obligation goes unnoticed, and impending renewals stay together. When you layer on Gatekeeper's AI, you can answer questions about any of them using all the knowledge in one record instead of siloed across teams.

Everyone wants to ask a question and get a correct answer, but without compiled, unified data, AI can still create false confidence. Gatekeeper gives you that confidence because it's unified data, and on top of Gatekeeper's AI, all 50+ agents are included, doing the manual work, taking action throughout the lifecycle so teams spend time making judgment calls rather than on admin.

Our MCP connectors allow you to connect Gatekeeper securely to any MCP-compatible AI assistant, so your team can work with live contract management and vendor data in their preferred AI tool. You don't have to choose between your current AI investments and Gatekeeper, meaning you can keep your chosen AI and work from a secure, specialist-built platform with data that isn't just stored but unified.

No More Surprises. Unlimited Users · Unlimited eSign · No AI Tax