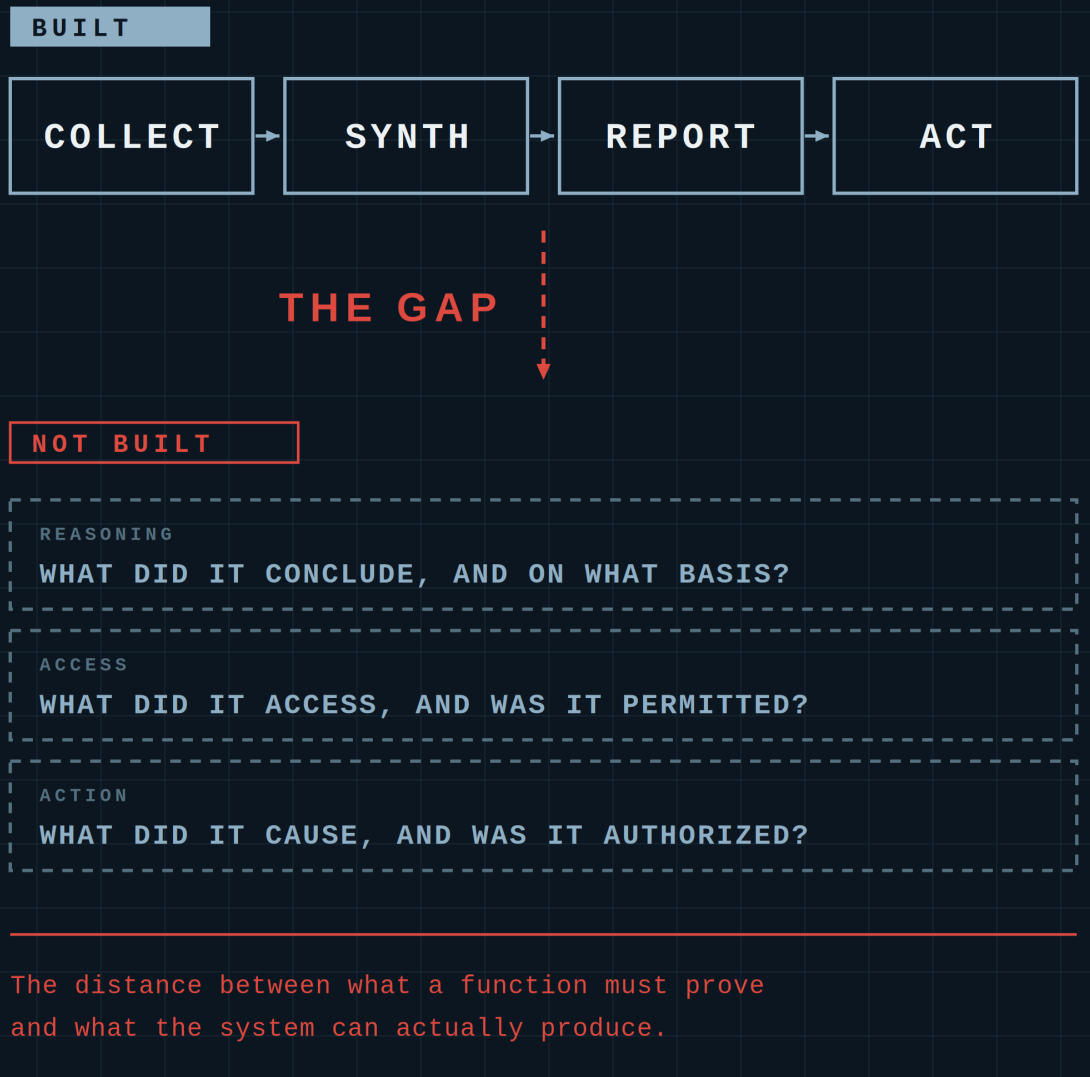
EVIDENCE DEBT

The system was designed
to operate.

Not to **prove** what it did.

By Stephen Alexander



The distance between what a function must prove
and what the system can actually produce.

AUTHOR'S NOTE

Two earlier essays argued that operational AI stops being a model problem and becomes a distributed-systems problem, and then tested that claim against a search-and-rescue mission.

In writing both, I kept circling the same question. How does a distributed operational AI system actually prove what it did? That it reached the right data, at the right time, for reasons it can show. That the conclusion it drew rests on something. That the action it took was one it was allowed to take, and that a human who could genuinely have said no was in a position to say it.

I do not mean the model's private reasoning, the weights. I mean each tool it called, each source it touched, each intermediate claim it made, each hand-off between the pieces, and every boundary where the record could quietly stop.

This essay goes directly at that.

These are my own views, not my employer's, and not a forecast from anyone's company.

The framework here is mine. I used an LLM to pressure-test it and to produce this as a designed PDF.

Companion work. cirrusly.me · sar.cirrusly-clever.com

THE ARGUMENT IN BRIEF

Technical debt bothers you immediately. Evidence debt does not. A system carrying a mountain of it ships, answers, and decides all day, and the debt sits there until someone asks a question it was never built to answer.

What a function must prove is fixed by the function itself, by what it touches and what it can cause, before anyone decides how to build it. That makes build, buy, borrow and SaaS into sourcing strategies rather than answers, and it makes the debt measurable before you ship.

Three questions, three dials, one ledger. And one case that no amount of working harder will close.

CONTENTS

> *Technical debt accumulates when we postpone engineering work.*

> *Evidence debt accumulates when we postpone building the ability to prove what an operational AI system reasoned, accessed, and caused.*

> *The system still works. That's what makes the debt so easy to ignore.*

Technical debt at least has the decency to bother you right away: it slows the next feature, complicates the next change, and you feel it every sprint. Evidence debt is the quiet kind. A system carrying a mountain of it ships, answers, makes its small decisions all day, and everyone moves on. The debt sits there, patient, until someone asks a question it was never built to answer.

Why did this decision happen? Who approved it? What data did it look at? Which model did the actual reasoning? Was it running somewhere we trust? Can you show that the agent never stepped outside what it was allowed to do?

None of those questions is hypothetical. Sooner or later something forces them: an auditor, a lawsuit, a regulator, a reporter, or the person on the other end who wants to know why. Any operational system that matters enough, runs long enough, or causes enough consequence eventually meets that day, and the only thing in your control is whether it can answer. Evidence debt is harder than the technical kind because you cannot pay it down after the fact: the proof had to be created at the moment the work happened. When it wasn't, it simply doesn't exist, and no scrambling later brings it back.

That gap has a source, a reason it hides, and a way to close it. It all turns on one idea: what a system must prove is set by the work it does, long before anyone picks the tools.

A COMMON PATTERN

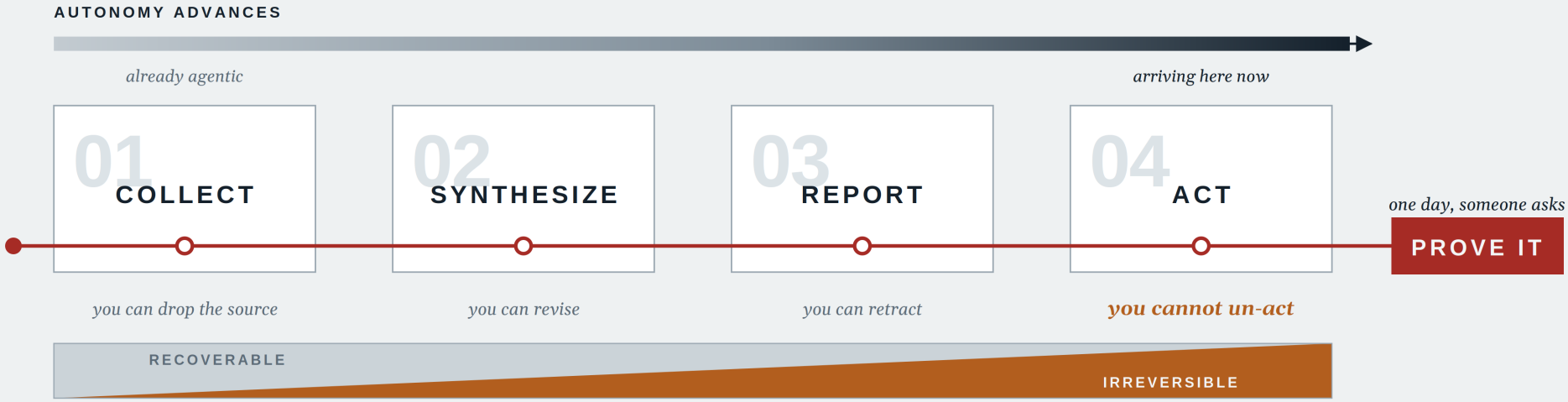
Much of the AI work being deployed right now follows a recognizable shape. Something collects data, synthesizes it into a conclusion, reports that conclusion to someone, and then someone acts on it. Collect, synthesize, report, act.

Two things give this pattern its stakes. Provenance: where each piece of data came from and whose hands it passed through, because a conclusion is only as good as the sources beneath it, and that lineage has to survive every step from collection to report. And reasoning: the basis for each choice along the way, what to gather, what to keep or drop, how to combine it, what to conclude, because a real decision waits at the end of the chain.

FIGURE 1

The four-step pattern

Evidence is created at the left. Proof is demanded at the right. The distance between them is evidence debt.



THE EXITS CLOSE

Each step spends one of your remaining chances to catch it. Autonomy is arriving where the exits have already shut.

THE THREAD IS THE EVIDENCE. IF IT BREAKS ANYWHERE ALONG THIS LINE, YOU CANNOT PROVE IT.

Once that shape is in your head, you see it everywhere, a bit like learning a new word and then hearing it three times before lunch. An investigative reporter gathers documents, reaches a conclusion, publishes, and defends it in public and sometimes in court. An investment firm runs the same loop and a capital decision follows. Same skeleton, different worlds, and in each, the thing challenged is the conclusion and the chain behind it.

The worked example here is OSINT security research, in the attribution-and-exposure sense: piecing together publicly and semi-publicly available information to reach a defensible conclusion about who is behind something, or what is dangerously exposed. This puts every part of the problem under pressure. The sources are messy, partly public, and some are planted to mislead you; the output can name a real person or organization; and getting it wrong can be close to permanent. If the ideas hold up here, they should travel well into the less adversarial cases too.

MOVEMENT ONE: A GOOD FIRST SKETCH

In an earlier essay, *Operational AI in Search and Rescue*, I drew a line between two things I called the inference substrate and the agentic substrate. Inference supplies the model execution, the raw work of turning an input into an output. The agentic substrate coordinates that execution into goal-directed work: actions in steps, a goal to chase, tools to call.

That line is a good one. It pulled apart two things people were mashing together, and let you talk about the model apart from the system that runs it. It is also not enough for the harder decisions a deployment turns on. Hold on to it anyway. It will earn its keep much later, when we have to decide which piece of a system moves.

Every operational function, sooner or later, has to answer three plain questions.

> *What did it conclude? What was it allowed to see? What did it ultimately cause?*

Those three name the obligations. Reasoning is what it concluded, and on what basis. Access is what it reached, and whether it was permitted to. Action is what it caused, and whether that was authorized.

FIGURE 2

The three questions every function gets asked

Each has two halves: what happened, and whether it was legitimate. How much evidence you need on each depends on the function.

	WHAT HAPPENED *what you captured*	WHETHER IT WAS LEGITIMATE *what you can conclude from it*
01 **REASONING**	*what it concluded*	*on what basis* *the sources, the model and prompt versions, the tool outputs, the human edits. Not the model's internal reasoning.*
02 **ACCESS**	*what data it reached*	*whether it was permitted to* *whether every source it touched was one it had permission to touch, for this purpose*
03 **ACTION**	*what it caused*	*whether it was authorized* *whether the agent could act on its own, or a named person held the decision and could genuinely refuse*

You capture records. Then you judge them against rules. A gap has two causes: you never captured the records, or you captured them and have no rules that say whether it was allowed. One costs instrumentation, the other costs policy. Both show up as an evidence debt.

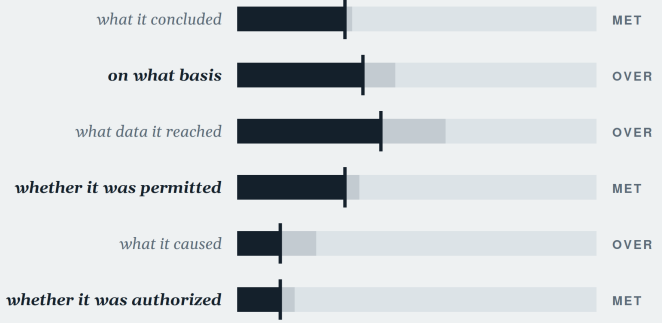
THE GAP, QUESTION BY QUESTION

Where the ticks are placed is how much evidence each question requires. Evidence debt does not net: a surplus on one question cannot pay a gap on another.

EVIDENCE PRODUCED | EVIDENCE REQUIRED | THE GAP | EXCESS. NOT CREDIT.

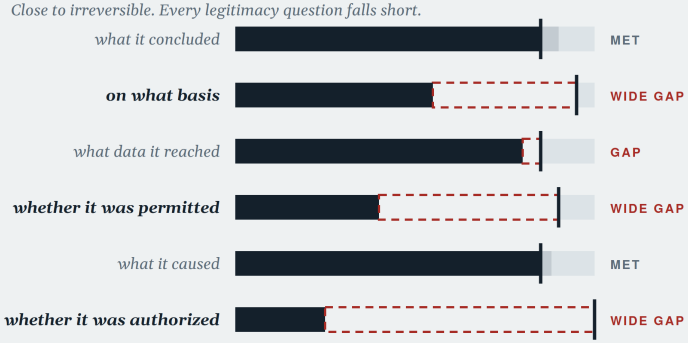
AN AUTOMATED SUMMARY FROM MANY SOURCES

Reviewed for accuracy and revised if needed. The consequence is recoverable.



NAMING A PERSON PUBLICLY

Close to irreversible. Every legitimacy question falls short.



Each legitimacy question depends on its own observed question. You cannot show that data access was permitted if you never recorded which data was accessed. If you did not capture enough, the legitimacy question cannot be fully answered.

Evidence debt is the gap between what a function must prove and what its implementation can actually produce. *Levels shown are illustrative.*

The two-substrate sketch describes what a function is built from. The three questions ask what it must prove. Those are different axes.

> *Evidence debt is the gap between what a function must prove and what its implementation can actually produce.*

MOVEMENT TWO: THE BREAKING POINT

In a real sourcing decision, the debt starts to grow.

Say you're standing up the security-research capability. You have three broad ways to get each piece. Buy a finished product and use it whole. Build it yourself, end to end. Or, where most real deployments land, buy a general-purpose agentic base and build custom pieces on top for the parts that touch your most sensitive work.

The two-substrate picture has no opinion about any of this. It cannot tell you whether the piece can answer the three questions when someone asks, and each sourcing choice opens the gap in its own way.

Buy the finished product, and you often get a clean, capable system with an opaque middle. Ask it to show its reasoning or which sources it looked at, and you may find nothing on offer, because the vendor built for output and left proof out. The function works. The evidence it requires is missing, and you won't find that out until the day you reach for it.

Build it yourself, and you would think you were safe, since you control the whole thing. Building a function gives you the ability to instrument it. It does not hand you the instrumentation for free. You get a record of reasoning, access, and action only where you deliberately wrote the capture in. A custom function nobody instrumented is exactly as silent as the opaque product you feared, and you paid more for the privilege.

The hybrid most teams choose, base plus custom, inherits both problems and adds a third: the seams. Every handoff from bought base to custom piece is where the evidence chain can quietly break, the reasoning on one side, the action on the other, and nothing stitching the record across the gap.

FIGURE 3a

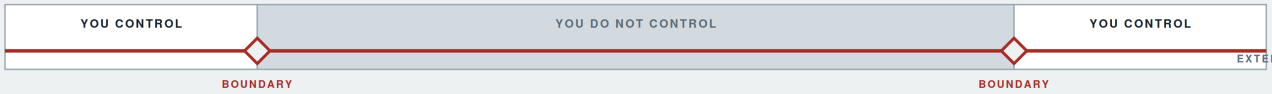
Every boundary is a decision point

A boundary is where control changes hands. Evidence may stop there, change form, or become inaccessible. It is a risk edge, and what happens at it is yours to decide.

THE EVIDENCE CHAIN: WHAT YOU CAN OBSERVE, AND WHAT THOSE OBSERVATIONS MEAN, CONTINUE ACROSS A BOUNDARY ONLY IF A MECHANISM CARRIES THEM.

01 USE A FINISHED PRODUCT OR SERVICE

A vendor product, or another team's service. Two edges either way.
Their records live inside their system. They are not automatically yours, and they do not automatically join to yours.



2 EXTERNAL CONTROL BOUNDARIES TO DECIDE ABOUT

02 BUILD AND OWN IT END TO END

One system you control, from intake to action.
No control changes hands. Hand off to an internal service you do not own and you have one again.

0 EXTERNAL CONTROL BOUNDARIES TO DECIDE ABOUT

03 ASSEMBLE IT FROM COMPONENTS UNDER DIFFERENT CONTROL

Bought, or borrowed from inside the house. It makes no difference.
Not worse. Just more decision points, and each one is easy to leave unmade.

4 EXTERNAL CONTROL BOUNDARIES TO DECIDE ABOUT

AT EVERY BOUNDARY, THREE THINGS CHANGE HANDS AND ONE DOES NOT

WHO OBSERVES
their instrumentation on one side, yours on the other

THE FORMAT
their schema, their identifiers, their clocks. Yours are different.

WHO JUDGES
their rules ran inside their system. Yours run in yours.

YOUR OBLIGATION REMAINS
On both sides, and at the line itself.

A BOUNDARY IS A RISK. TREAT IT LIKE ONE.

AVOID
Remove the boundary. Own the whole path, or consume the whole thing. Fewer joins, fewer places to fail.

REDUCE
Carry observing and judging across. A shared identifier, a record of the crossing, a contract on what the interface hands over. It costs far less at design time than in reconstruction.

TRANSFER
Make them expose it, by contract or by internal agreement. You can transfer the work. You cannot transfer the obligation, so this is only ever partial.

ACCEPT
For a low-stakes function this may be entirely fine. Accepting a gap knowingly is a decision. Finding it in a deposition is a debt.

Deciding well does not guarantee the evidence survives. Schemas drift, retention lapses, controls fail. The debt starts earlier than any of that.

Evidence debt grows at every boundary where no mechanism carries the observing and the judging across.

FIGURE 3b

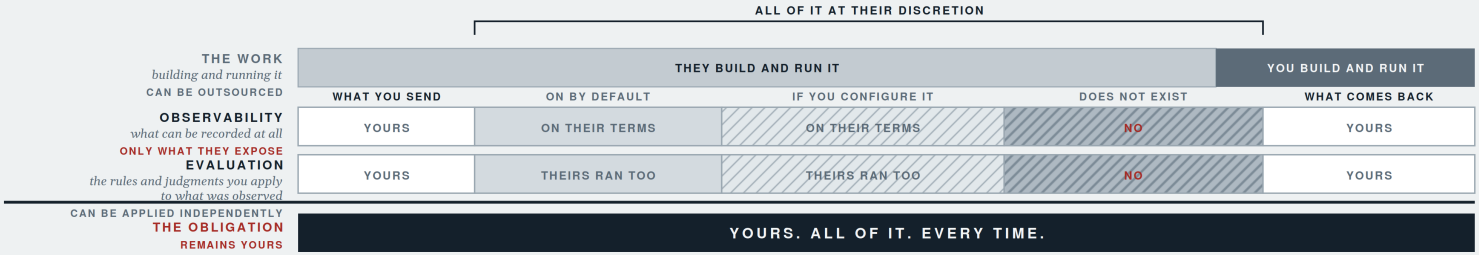
You can outsource the work, not the obligation.

Control can be divided across products, teams, and vendors. The evidence requirements stay with the function.

YOU CONTROL IT. | THEIRS: ON BY DEFAULT. | THEIRS: IF YOU CONFIGURE IT. | THEIRS: DOES NOT EXIST.

01 USE A FINISHED PRODUCT OR SERVICE

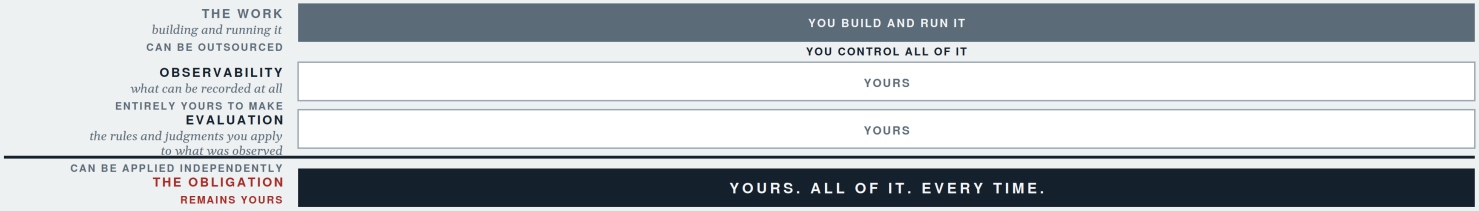
You hold both ends. They control the middle.



Some of their evidence is on by default. Some you can switch on. Some does not exist at any price. All three are theirs to decide.

02 BUILD AND OWN IT END TO END

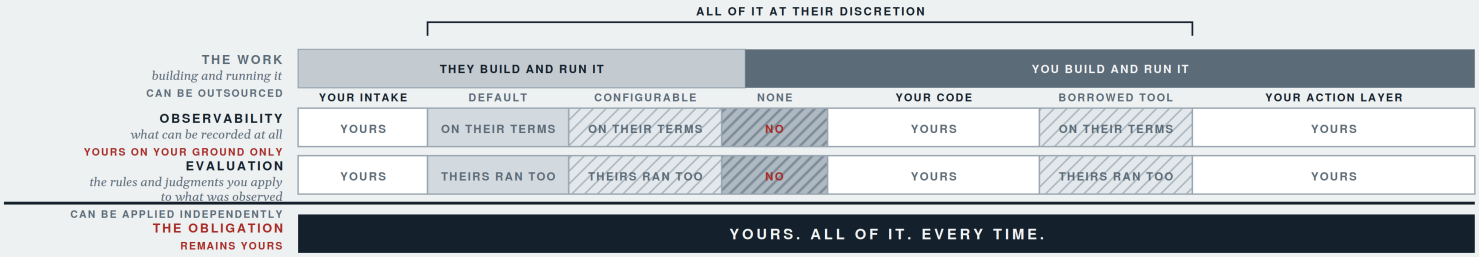
One system you control, from intake to action.



Nothing is sealed. Every observation can be made available and every rule can be applied. None of it exists until you build it, run it, and keep it running.

03 ASSEMBLE IT FROM COMPONENTS UNDER DIFFERENT CONTROL

Bought, or borrowed from inside the house.



You control some ground and rent the rest. Control is divided. The obligation remains yours.

Rules without observations cannot establish anything. Observations inside someone else's system exist only if they choose to expose them.
So you can be obligated to establish something you have no way to establish. That is what makes this debt different: you cannot always pay it down by working harder.

Evidence debt begins when a function must prove something that no one in the delivery chain can produce.

FIGURE 3c

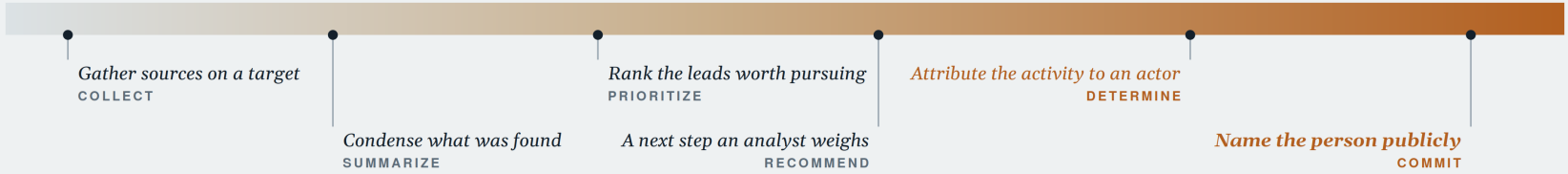
What the function requires decides what you can source

Evidence requirements belong to the function. They set how much control you have to keep, and how much boundary risk you can accept.

THE TEST: FOR THIS FUNCTION, IS EVIDENCE UNDIFFERENTIATED WORK, OR IS IT THE DIFFERENTIATED CORE?

EVIDENCE IS UNDIFFERENTIATED WORK
you gain nothing by owning it

EVIDENCE IS THE DIFFERENTIATED CORE
the defensibility of the conclusion is the value



THE SAME FOUR STEPS FROM FIGURE 1

COLLECT	SYNTHESIZE	REPORT	ACT

In this example, Collect sits at the low end because errors are recoverable, source access is observable, and acceptable-source rules are straightforward to enforce. That makes it a good candidate for outsourcing.

AND THAT DECIDES HOW MUCH CONTROL YOU HAVE TO KEEP

	LOW REQUIREMENT	HIGH REQUIREMENT
SOURCING	*A finished product or service is fine. Take what they expose.*	*Require sufficient observability wherever evidence is required. In systems you do not control, that depends on what the provider exposes.*
BOUNDARIES (3a)	*Accept them. A boundary you knowingly accept is a decision, not a debt.*	*Avoid or reduce them. Every untreated boundary can widen the gap between what a function must prove and the evidence the delivery chain can produce.*
THE OBLIGATION (3b)	*Yours. But it is small, and what they expose covers it.*	*Yours. Others may produce evidence on your behalf. They do not change what the function must prove.*

THE TRAP *Evidence infrastructure can look like commodity plumbing. That makes it easy to outsource before asking whether the evidence it produces is itself part of the product.*

Sourcing is not the first decision. What the function must prove is the first decision. Sourcing is what you choose once you know the answer.

Evidence debt is the gap between what the function must prove and what the chosen sourcing model can actually produce.

The steps shown are this essay's example. The rungs beneath them are general. Placements are illustrative.

The clean picture that felt like understanding doesn't tell you what to build, and while you decided how, evidence debt accrued on every function you shipped. It costs nothing yet, which is the whole problem. The cost is real, it is mounting, and it stays invisible until someone asks a question and the silence answers for them.

MOVEMENT THREE: A SHARPER RULE

> *Evidence is a property of the function, not the implementation.*

What a function has to prove is fixed by the function itself, by what it touches and what it can cause, before anyone decides how to build it. A function that names a real person must give a heavy account of its reasoning and sources, whatever stack it runs on. A function that drafts a throwaway summary must prove very little. The requirement comes from the job, waiting there before the build-versus-buy conversation starts.

That move turns build, buy, borrow, open source, and SaaS into sourcing strategies, different ways of meeting a requirement the function already handed you. It also makes evidence debt measurable in advance: with the requirement fixed, you can read the gap between what a function must prove and what your build produces before you ship, ahead of the crisis where you would otherwise find it.

The discipline is this: before you ask how to build a function, ask what it must prove, then choose a way to build it that can prove that. Call it evidence sufficiency, keeping the gap at zero on purpose.

The triad also tells us what the system must produce. If reasoning, access, and action are the three things a function must prove, the system needs something that produces each. Gather those producers and you have the evidence plane, the part of your system whose whole job is to generate proof.

The plane is not a product, and not a box in your architecture. It is a role, played by parts you mostly already have: identity, logs, policy, lineage, and some that do not exist yet. Each producer is a function in its own right, and the same three questions get asked of it. Three parts produce the three answers, and a fourth reaches underneath all of them. Read each as the primary producer of its evidence, not the only one. The split is a working theory. The triad under it is the load-bearing idea.

Identity and data access handle access evidence: who or what reached which data, under what permission, and whether the access stayed inside the limits set for that purpose.

Policy, authority, and execution records handle action evidence: what a function was permitted to cause, and the record of what it actually did.

Audit and observability handle reasoning: the trail of how a conclusion was reached. This does not mean recovering a model's private internal reasoning. It means preserving the observable basis of the conclusion: the provenance of each source, the model and prompt versions, the tool outputs, the intermediate claims, and any human interventions. Provenance sits at the center, which is fitting, since provenance is what the whole pattern rested on from the first paragraph.

Each producer keeps a chain that has to reach the end intact: provenance for reasoning, delegation for access, authority for action. Every one of them breaks at a boundary of control, which is the next thing to look at.

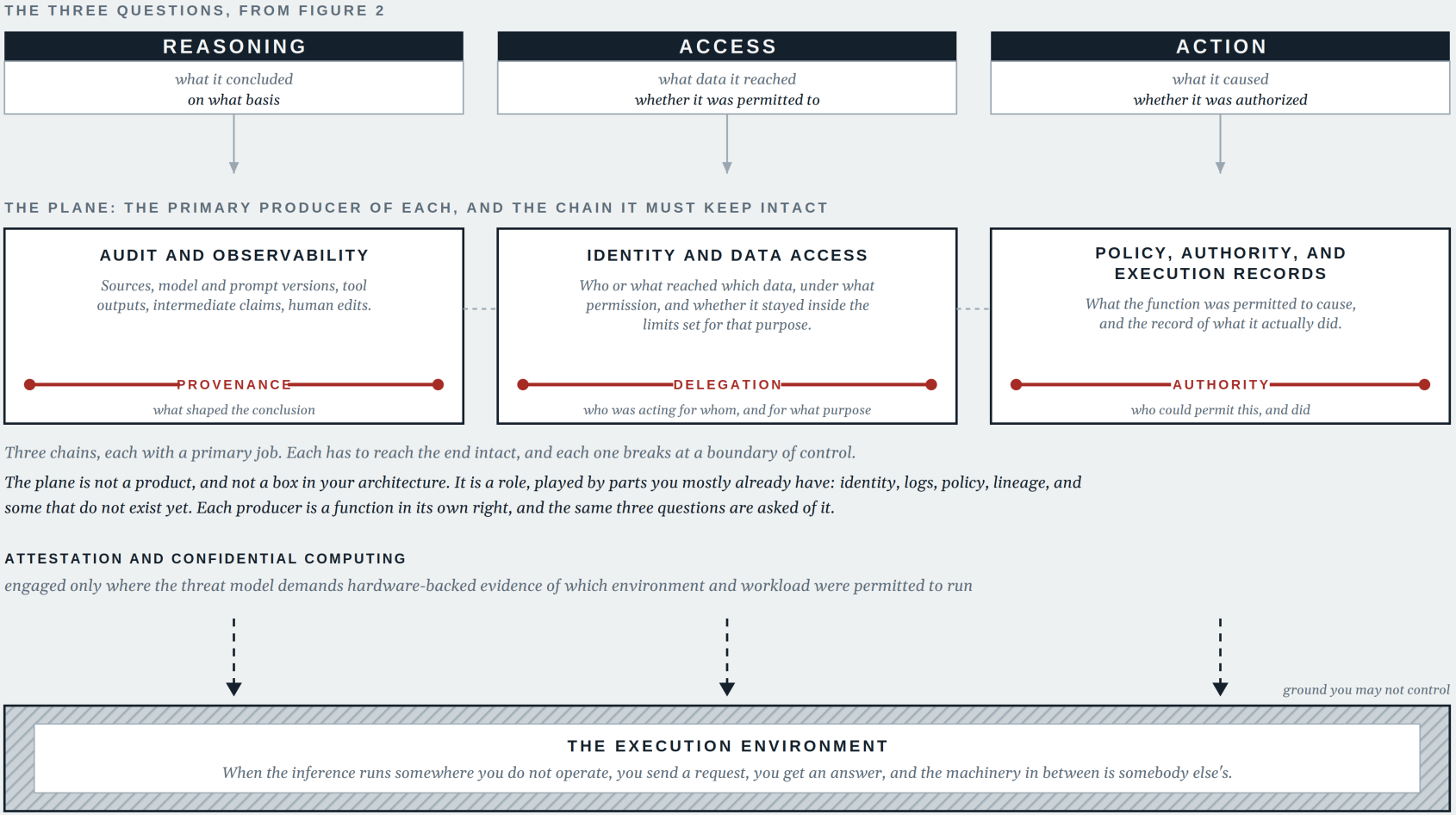
The fourth part reaches down to the execution environment itself. Inference is something you call: you send a request and get an answer back, and the machinery in between is somebody else's. Confidential computing runs the work inside a hardware-isolated environment built to protect data and code in use, even from the operators, with guarantees that hold only as far as the threat model and implementation do; attestation then gives cryptographic evidence about which environment

and workload were allowed to run. Attestation is not a foundation under the plane. It is a reach downward, taken where the threat model requires it, and for some functions it should be. In 2026 it is hard: proving which environment and workload ran is well understood, but doing it for a whole agentic loop is not, because every hop you leave outside is another crossing to attest, and nothing yet stitches them into one record.

FIGURE 4

The evidence plane

The three questions set what the function must prove. The plane is everything in your system that does the proving.



Three chains, each with a primary job. Each has to reach the end intact, and each one breaks at a boundary of control.

The plane is not a product, and not a box in your architecture. It is a role, played by parts you mostly already have: identity, logs, policy, lineage, and some that do not exist yet. Each producer is a function in its own right, and the same three questions are asked of it.

Attestation is a reach downward, not a foundation. It can be done, and for some functions it should be. In 2026 it is hard: bind it across an agentic workflow, its inputs and its tool calls, and every hop multiplies the boundary you attest. Performance costs are unsettled.

Where you cannot close a gap today, record it, size it, and schedule it. An unrecorded gap is the same risk, just one nobody is managing.

Evidence debt is a chain the plane cannot keep intact.

The requirement side resolves into a gradient set by three dials, and they are set separately: how the material is classified, what it costs to undo the act, and who can compel the basis for the conclusion. The same function climbs through three increasingly demanding settings, configurations of one example rather than a universal three-tier model.

FIGURE 5

One function, three configurations

The dials decide what a function must prove. They are set separately, and they do not climb together.

THREE DIALS, SET SEPARATELY

Restricted: shared under terms, an NDA, a licence, a purchase. Protected: PII, PHI, privileged, regulated in its own right.
Contractual: an internal standard, an SLA or a client can demand the basis. Regulatory or legal: a regulator or a court can compel it.

THE SAME SECURITY-RESEARCH FUNCTION, AT THREE SETTINGS

Each row sets the same three dials, in the same order as above.

	THE DIALS, SET	WHAT IT DOES	WHAT IT MUST PROVE	WEIGHT
01 **ADVISORY** *a research assistant* 3c RUNGS: COLLECT, SUMMARIZE		*Gathers open material into a dossier of what exists and where it came from. It concludes nothing. The analyst does the weighing.*	What it collected, and from where. Enough to retrace. A bad source is a quality problem the analyst catches.	
02 **HIGH ASSURANCE** *it draws its own conclusions* 3c RUNG: DETERMINE		*Links accounts, matches a technique to earlier activity, and asserts that scattered signals point to an actor. The sources are still open.*	Provenance of every source under the conclusion. The analytic path to the claim. Some way to tell whether an input was planted to fool you. ***Without this, it cannot yet run on its own.***	
03 **CLOSED BOUNDARY** *it can name someone* 3c RUNG: COMMIT		*Names a person or an organization publicly, or feeds a conclusion into a decision that changes a life. You cannot un-name someone.*	Everything above, plus proof the agent held no unilateral authority, that a named human saw the evidence and could genuinely refuse, and that the act was taken under that authority.	

Consequence is the only dial that climbs through all three. The sources stay open until the last step, and someone could always ask for the basis.
Three dials, because they do not move together. A human holds the naming because the act cannot be undone, not because the material is protected.

THE SAME THREE DIALS, ON TWO OTHER FUNCTIONS

SEARCH A SEALED ARCHIVE		*Nothing acts on it and errors are recoverable, so consequence sits at the floor. The material is protected, so a regulator can still compel every access. Sensitivity alone is what brings the model to the data. Figure 6.*
TAKE DOWN A PUBLIC POST		*The post is public, so sensitivity sits at the floor. You can put it back up, but you cannot un-suppress it, and a wrongful takedown is answerable in law.*

Dials move after you ship, and they mostly move up: the output that informed last quarter is acted on this quarter, a rule arrives, a client adds an audit clause.
Sometimes the dial never moved at all, and you read it wrong at the start. Either way, the evidence you can produce does not rise to meet it on its own.

Evidence debt is a dial sitting higher than the evidence you can produce.

On the first step, advisory, the function is a research assistant. It gathers and organizes open material into a dossier of what exists and where it came from. It concludes nothing on its own; a human analyst does the weighing. The evidence it must produce is light: a record of what it collected and from where, enough to retrace later. A bad source or a fumbled label is a quality issue the analyst catches, and it stays clear of the evidence question.

On the second step, high assurance, the function draws conclusions of its own. It links accounts, matches a technique to earlier known activity, and asserts that scattered signals point to a particular actor or exposure. Now it is producing a claim someone might act on, built partly on data others have deliberately salted with falsehoods. The bar jumps hard. You need the provenance of every source the conclusion rests on, the analytic path to the claim, and some way to tell whether an input was planted to fool you. Provenance stops being tidy hygiene here and becomes the entire case. If you can't produce that evidence, the honest answer today is that the function isn't ready to run on its own yet.

MOVEMENT FOUR: THE WORST CASE

The third step is a closed boundary. The function can now name a person or an organization publicly, or feed a conclusion into a decision that materially changes someone's life. That is about as close to irreversible as this work gets. You can claw back a lot of mistakes in this world. You cannot un-name someone.

And here the dials come apart, which is the whole reason there are three of them. Watch sensitivity: it barely moves. This is open-source research, so the material was public all along, and even here it climbs only to restricted. What went to the ceiling is consequence, because the act cannot be undone, and contestability, because a court can now compel the basis for it. Two dials at the top, one near the floor. If they always climbed together they would be one dial.

This step is the test: the worst case for the framework. If the rule holds here, at maximum stakes and irreversibility, it holds everywhere below. If it needs some new idea bolted on to cope, it was never as general as I claimed.

The old sketch gives you nothing here. Whether this function is "inference or agentic," whether you build it or buy it, both questions slide off the real problem, which is about what must be proven. The evidence rule produces two consequences a weaker framing can't reach, and they do not come from the same dial. The first is driven by consequence. The second is driven by sensitivity, and it is the reason the two have to be told apart.

The first consequence flips the evidence you collect. All the way up the gradient you were proving what the agent did. At this step a class of decisions becomes non-delegable by rule: a human takes authority over the naming and the exposure, while the agent's job narrows to assembling the dossier the human judges. The thing you prove inverts, and proof of what the agent refrained from doing becomes the artifact. Concretely, the evidence must show the agent lacked unilateral authority for the final call, that the workflow required an informed human decision, and that the consequential action was taken under that authority.

And it has to be real proof, which means something uncomfortable about the human here. Real oversight means the human sees the evidence before the recommendation and holds an actual ability to refuse, to look at the dossier and say "this isn't good enough, we don't publish." A human who only sees the final recommendation and clicks approve is a rubber stamp wearing a lanyard, and worse, launders responsibility onto a person who never had a real chance to say no. The evidence required here includes proof that the refusal was both possible and informed.

> *The second consequence changes where the work happens, and it is not the naming that triggers it. It is the material.*

Every time work crosses a boundary, someone on the other side decides what gets recorded about it. Often they already record what the function requires. Often you can configure and contract your way to it. But sometimes their capability stops short, and no configuration, contract or effort writes a record their system never kept. That is the case nothing else here can fix, and protected material is how you usually land in it.

So change what crosses. You cannot delete a boundary, and something always has to cross it. Send the material out and you send the thing you cannot make a list of: every record, every query, and more arriving tomorrow, each access another crossing to account for. Send the work inward and you send the thing you can list: one environment, one workload, one configuration, which you can name, pin, and get a receipt for. The material stays where it is already permitted to sit. This is the closed-boundary inversion.

And here the two-substrate sketch finally earns its keep. It could never tell you what a function must prove. It tells you exactly what to move: the inference substrate, and every part of the agentic substrate that touches the material. Move the inference alone and the orchestrator, the retrieval and the tool calls still cross, so the material crosses with them. The boundary is reduced, not removed, and a reduced boundary is dangerous mostly because it looks like a removed one.

The environment does not have to be yours; an accredited provider can hold it. You can transfer the work, never the obligation. And moving the work does not write the record, any more than building a function did. It gives you the ability to instrument, and nobody does that part for you either.

FIGURE 6

Change what crosses the boundary

You cannot delete a boundary. Something always crosses it. So decide what.

WHERE THIS PICKS UP

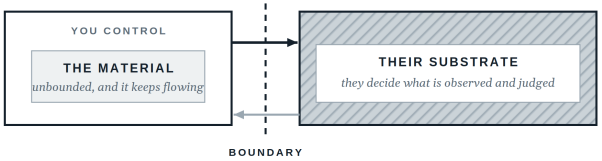
ON BY DEFAULT | IF YOU CONFIGURE IT | THIS ONE: NOT ON OFFER

Figures 3a and 3b left three cases. Two are workable. This is the third, and it is the one 3b could not resolve.

SENSITIVITY
how you usually get here

01 THE MATERIAL CROSSES

The ordinary design. It works wherever the first two cases apply.



WHAT YOU CAN GET FROM THE OTHER SIDE

You may already get everything the function requires: their records and their rules, on by default. You may be able to reach it by turning things on and writing it into the contract. Or their capability may stop short, and nothing you configure or sign closes the distance.

Find out which, before you rely on it. A gap you accept knowingly is a decision. A gap you never checked for is a debt.

02 THE SUBSTRATE CROSSES INSTEAD

The inference substrate, and every part of the agentic substrate that touches the material.

ATTESTATION — *evidence the move really happened: which environment, which workload*

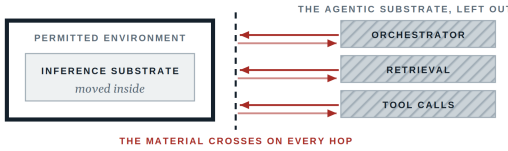
WHY THIS WORKS

You cannot make a list of the material. It is every record and every query, and more of it arrives tomorrow. Every access to it is another crossing. You can make a list of the substrate. One environment, one workload, one configuration. Name it, pin it, and get a receipt saying that is what ran.

The trade: you hold a permitted environment and trust a workload you did not build, and hundreds of crossings become one you can instrument.

The two-substrate sketch could not tell you what a function must prove. **It tells you exactly what to move.**
Inference substrate, plus every part of the agentic substrate that touches the material. That is the unit that relocates.

THE PARTIAL MOVE, AND THE DEBT IT HIDES



Moving the inference substrate solves part of the problem.

It relocates where the material is processed, and that is real. The orchestrator, the retrieval and the tool calls still cross, so the material crosses with them, and those hops sit in whichever of the three cases they were always in.

The boundary is reduced, which is 3a's REDUCE and a real outcome. Removing it is AVOID, and that takes the whole loop. The risk is that a reduced boundary looks like a removed one.

WHAT THE MOVE DOES NOT DO

IT DOES NOT WRITE THE RECORD
You get the ability to instrument. Instrumentation is a separate job, and nobody does it for you.

IT DOES NOT MEAN YOU RUN IT
An accredited provider can hold the environment. You can transfer the work. The obligation stays yours. (3b)

IT DOES NOT REMOVE BOUNDARIES
You take the workload image on trust. Attestation proves which workload ran, not that it deserved to. (3a)

IT DOES NOT PROVE THE ANSWER
Where it ran is not why it is true. Attestation is evidence about the environment, not about the claim.

Attestation is difficult. Proving which environment and workload ran is well understood. Doing it for a whole agentic loop is not, because every hop you leave outside is another crossing to attest, and nothing yet stitches them into one record.

This move is the closed-boundary inversion.

Some evidence debt is architectural. You do not work it off. You remove it by changing what crosses the boundary.

Handling the worst case took no new machinery. The same rule that governed the low-stakes dossier on the first step is the rule that produced "prove the agent didn't act" and "change what crosses the boundary" at the top. It held here, so it holds everywhere below.

WHERE THIS LEAVES YOU

So put the evidence question ahead of the build conversation, per function: what must this prove about its reasoning, access, and action, and can the way you'll build it produce that proof? The answers sort your work. The light steps lean on whatever is easiest to source; the heavy ones get the real instrumentation, the closed environments, the human who can actually refuse.

Make that concrete with a ledger, one row per function. Its columns: the function, the reasoning, access, and action evidence it must prove, what the build actually produces, and the gap. That gap column is your evidence debt, function by function, in a form you can read on a quiet Tuesday instead of discovering in a deposition. In practice, give the gap a severity, from none to low to material to blocking, and bind each one to a deployment decision you have made in advance. A severity that does not bind a decision is a colour on a chart.

FIGURE 7

The evidence ledger

One row per function. What it must prove, what the build produces, and the distance between them.

FUNCTION	REASONING	ACCESS	ACTION	WHAT THE BUILD PRODUCES	THE GAP
01 **DRAFT A SOURCE DOSSIER** *advisory. Nothing acts on it.*	**Trail of sources gathered.** *Enough to retrace what was collected, and from where.*	**Which sources were touched.** *A record of what was reached, and what was only attempted.*	**None. Advisory only.** *The analyst does the weighing. The function concludes nothing.*	**Collection log.** *A log of sources collected and attempted.*	LOW
02 **ATTRIBUTE AN ACTOR** *from raw material to a claim.*	**Analytic path, and planted-input checks.** *The steps to the claim, and a way to tell if an input was planted.*	**Permitted sources, with provenance.** *What you had the right to see, and where each piece came from.*	**Claim recorded, basis shown.** *The claim, tied to the evidence underneath it.*	**Partial.** *Some of it is visible. The links that carry the claim are not.*	MATERIAL
03 **NAME A SUBJECT PUBLICLY** *irreversible, and contested.*	**A full defensible chain.** *End to end, and able to hold up when someone comes at it.*	**Provenance, and confidentiality on sources.** *Where every piece came from, and how the sensitive ones were handled.*	**Proof a human held authority.** *That the agent could not act alone, and a named person saw the evidence and could refuse.*	**Often missing.** *The evidence rarely exists end to end today.*	BLOCKING

PRODUCED
REQUIRED
THE GAP

WHAT EACH SEVERITY BINDS YOU TO

NONE	LOW	MATERIAL	BLOCKING
Ship it.	Ship it, and write the gap down where someone will see it.	Do not run it unattended. Close the gap, or keep a named human in authority.	Do not deploy this function until the gap is closed.

If you cannot fill the gap column for a function, evidence debt was never measurable for it, and learning that early is a gift.

Evidence debt is measurable before you ship. This is the measurement.

The table is the whole argument made operational, which keeps the measurability claim honest. If you can't fill the gap column for a function, evidence debt was never really measurable for it, and learning that early is a gift.

One timing wrinkle: the tooling is young, and today it mostly won't hand you attestation or end-to-end evidence capture. That is why asking the evidence question now matters, and why your heaviest functions often have to be built in-house to clear the bar today. Expect that to change, and build for the migration, so you can move to a bought option later without tearing out your proof.

That ledger is the thing to keep. It holds, per function, what the work requires and what the build can prove, and the space between is a number you would rather see now than discover later. Most teams carry more of that debt than they realize, quietly, on systems that run just fine.

Most of it comes down to instrumentation, and instrumentation is work you can do, as long as you do it at the moment the work happens, which is the only moment you can. Some of it will not come down that way. Some evidence debt is architectural, and no amount of working harder touches it; you remove that kind by changing what crosses the boundary. Either way, the debt is only ever payable in advance. Nobody has ever built the evidence afterwards, because there was nothing to build it from.