SINGULARICS

White Paper

Knowledge Investment

Funding the Domain Knowledge Network.
Was implicit. Now load-bearing.

Alexander S. Petty
SINGULARICS | singularics.ai

First published August 2026.

singularics.ai

Contents

Abstract

Domain knowledge has always been essential. What changed is that it used to be a given and is now the constraint. When execution was scarce, organizations organized around the people who could build. The domain experts who knew what to build and whether the result was right were valuable but structurally invisible. Their time was not budgeted. Their contributions were not encoded. Their availability was assumed rather than planned. That worked when delivery was the bottleneck, because the slow delivery cadence never outran the supply of judgment. It stops working the moment delivery becomes cheap and fast, because every daily cycle needs domain context that must arrive before the build starts and judgment that must land before the day ends. This paper makes the case for treating domain knowledge as a funded, measured, compounding investment rather than as overhead that happens to exist.

Knowledge Investment, Now Explicit

The first time we ran an AI-native team against a regulated financial workflow, the fleet produced a complete implementation in two days. It took three more days to discover that the implementation applied the wrong day-count convention for accrued interest, a distinction that lives in one person's head and zero documents. The rework cost five build cycles. The encoding that would have prevented it cost thirty minutes. That ratio, thirty minutes of prevention versus five days of rework, is the business case for knowledge investment in a single data point.

For decades, domain expertise was a background condition. Organizations hired underwriters, credit analysts, compliance officers, and policy specialists, and those people contributed their knowledge through meetings, reviews, and hallway conversations. Nobody budgeted their time as a network resource. Nobody measured whether their contributions outlived the conversation. It worked well enough, because the delivery cadence was slow enough that the supply of domain judgment never became the constraint.

That arrangement breaks when the cost of an increment falls toward zero. The constraint does not disappear. It moves to the slowest remaining human step, which is deciding what should be built, judging whether what came back is right, and supplying the domain knowledge needed to make either call. Human judgment becomes the bottleneck.

> *When execution was the constraint, domain knowledge was a given. Now that execution is cheap, domain context and timely judgment are the scarce resources. Scarce resources need funded structure.*

An AI-native operating model runs a daily cycle: specify, build, judge, land. Every turn of that cycle needs domain context injected before the build starts and domain judgment applied before the day ends. If that context arrives late, the fleet builds against thin knowledge and the Decision Owner judges against incomplete evidence. If it does not arrive at all, the cycle either stalls or produces work that looks right and is not.

The Domain Knowledge Network is the structure that ensures context arrives on time and judgment is available when it is needed. But a structure without funding is a suggestion. Treating the network as volunteer work, something experts do on top of their real jobs, produces a network that works when people have time and fails when they do not. The binding constraint becomes the one that is resourced least.

Making knowledge investment explicit means three things. First, the time that experts spend on network obligations is budgeted, not borrowed. Second, the knowledge they contribute is encoded into the harness and the knowledge graph, not merely explained to the people in the room. Third, the health of the network is measured the same way any other investment is measured: by whether its returns compound.

Invisible Domain Contribution

The practice this replaces is structurally invisible domain contribution. In the old model, a domain expert answered questions when someone found them. A compliance officer reviewed a document when someone sent it over. A policy analyst corrected an interpretation when someone got it wrong. None of this was budgeted, planned, or measured. It happened through relationships and availability.

That worked when delivery cadences were slow. A two-week sprint could absorb a two-day wait for an expert. A quarterly release could absorb a month of accumulated misunderstandings that were corrected during acceptance testing.

A daily cycle cannot absorb either. If the expert's input does not arrive before the build starts, the fleet builds against thin knowledge. If the expert's validation does not arrive before the day ends, the Decision Owner judges against incomplete

evidence. The old model of ad hoc, unfunded, unplanned expert contribution does not fail gracefully in a daily cycle. It fails completely.

Knowledge investment also replaces the organizational fiction that business analysts are a sufficient proxy for domain expertise. A business analyst can translate between business and technology. A domain expert can tell you whether the result is correct against policy, regulation, and institutional precedent. These are different capabilities, and the daily cycle needs both. The Domain Knowledge Network organizes that expertise so it can be injected, encoded, and reused.

The One Rule and the First Line

Every expert floats and none are dedicated. That is the one rule of the Domain Knowledge Network, and it has exactly one exception.

The network's guild of policy, compliance, risk, operations, and data experts goes where knowledge is needed today. No expert is permanently assigned to a team. They rotate on a schedule the Flow Council sets at weekly calibration, moving to where the current week's specs require their specialty. This is what makes a finite guild viable across a growing portfolio.

The exception is the Decision Owner. The Decision Owner is drawn from this same business guild and sits with one team full time, at 60% or more of their working week. Because the Decision Owner is embedded, the team already holds the domain knowledge it needs for day-to-day decisions. That is the point of embedding them.

> *The Decision Owner is the first line of domain knowledge. If a team is calling the network for questions its own Decision Owner could answer, the wrong person was named as owner. That is a naming failure rather than a network failure, and it is fixed at formation, not by adding more experts.*

The network exists for reach beyond what the owner carries. An outcome owned by a pricing expert that runs into a disclosure rule needs a specialist the owner is not. An operations outcome that hits a policy constraint needs the compliance expert. The network provides that reach on a push model: it connects the right expert to the right team before anyone has to ask.

Push vs. Pull: How Injection Works

The distinction between push and pull is the difference between a network that works and one that stalls teams.

Pull model (what this replaces). The team discovers it needs domain knowledge mid-build. Someone sends a message to the expert. The expert is busy. Four hours pass. The Fleet Lead is blocked. The build day produces nothing usable. The expert answers the next morning, and the team has lost a full cycle.

Push model (what this funds). Each evening, daily planning names the expert tomorrow's spec will need. The network reads the signal and connects the right expert to the right team before the build starts. A typical contribution is a 15 to 30 minute working session first thing in the morning. The expert hands over the policy constraints and validation criteria the work must satisfy. The Fleet Lead has what they need before the agents start. No team stalls waiting.

The signal mechanism is daily planning, which is why daily planning's output must include a named domain expert for the next day's work. If daily planning does not produce that signal, the push model has no trigger, and the network reverts to pull whether anyone intended it to or not.

15–30 min per injection session. The expert arrives with the specific constraint knowledge the spec needs. This is a working session, not a review cycle. Governance arrives as knowledge on the way in, rather than as a review queue on the way out.

The Funded Lines

Knowledge investment is not a single line item. It funds four things, and skipping any one of them breaks the return on the others.

Network membership

Members of the Domain Knowledge Network are drawn from policy, compliance, risk, operations, data, and the business analysts who hold institutional knowledge. They remain in their home functions. Network membership is an additional, named, funded commitment of typically 20 to 30% of their week.

20–30% of each expert's working week is funded for network obligations. This is not volunteer time. It is a named commitment with the same standing as any other allocation.

That percentage covers the time they spend injecting context into daily cycles, validating specs, encoding their criteria into the harness, and maintaining the knowledge graph. It does not cover the time they spend in their home function doing their primary job. The two are distinct, and treating network time as a subset of the expert's existing workload is how the network becomes the first thing cut when the home function gets busy.

Monthly network sessions

Budget a two-hour session once a month for the full network. The session has a specific structure:

Hour one: absorption. Questions, edge cases, and new patterns that flowed back from teams during the month are reviewed. Each item is either encoded into the harness and knowledge graph during the session or assigned to a named owner with a deadline. Nothing leaves hour one as an open item.

Hour two: revision. The playbooks and context sets that teams will receive next month are updated based on what was absorbed. Stale entries in the knowledge graph are flagged and corrected. New checks written during hour one are reviewed for accuracy by a second expert.

Skip this session and the network becomes a rota of interruptions. Experts answer the same questions month after month because nothing was encoded. The session is where individual contributions become shared knowledge, and shared knowledge is what makes the guild scalable.

The session is not a meeting where people listen to a presentation. It is a working session where artifacts are produced. If the two hours end and nothing was encoded, the session failed, no matter how productive the conversation felt.

Knowledge graph maintenance

The knowledge graph is the structured record of domain context, policy constraints, decision precedents, and validation criteria that the network has accumulated. It is what the spec agent draws on when drafting criteria, what the fleet receives as working context, and what the harness checks against.

A knowledge graph that is not maintained degrades. Policies change. Precedents are overturned. Edge cases are discovered. Without regular maintenance, the

graph drifts from reality and the fleet works from stale context. Fund maintenance as a standing obligation, not as a project with an end date.

Decision Owner training in context engineering

Decision Owners are drawn from the same business guild as the network, and they need a specific skill: context engineering fluency. That means knowing what the fleet needs to be told, writing criteria that are checkable, curating the context set, and reading agent output critically.

Build it by pairing a new Decision Owner with an experienced Fleet Lead for their first two weeks of spec sessions. It transfers by apprenticeship, not by training course. Two weeks of pairing produces a competent context engineer. A two-day training course produces someone who has heard the vocabulary.

Fund the pairing. The Fleet Lead's time during those two weeks is a knowledge investment, and it pays back every day the Decision Owner closes cycles without the fleet stalling on ambiguity.

The Network as a Funded Commitment

The single most important sentence about the Domain Knowledge Network is this: it is not volunteer work.

> *Named, funded, measured. Members stay in their home functions, but network obligations are real. If membership competes with the home function for the same hours and the home function always wins, you do not have a network. You have a list of names.*

The distinction does real work because knowledge networks fail silently. There is no red build, no missed deadline, no blocker that shows up in a stand-up. What happens instead is that context arrives thin, specs miss constraints, and the harness does not cover the policy that changed last quarter. The quality degradation is invisible until a production incident or an audit finding surfaces it, and by then the cost of repair is orders of magnitude higher than the cost of the contribution that was skipped.

Funded, in practice

Three things make the difference between a funded commitment and an aspiration.

The time is allocated by the home function's leadership. The sponsoring manager agrees to release the expert for 20 to 30% of their week, and that agreement is recorded the same way a Decision Owner's 60% release is recorded. It is not a handshake. It is a capacity commitment.

The obligations are named. Each network member has a defined rotation that the Flow Council sets at weekly calibration. They know which teams will need them this week, which specs they will validate, and which knowledge graph entries they own. The work is planned alongside capacity, not begged for in the moment.

The contribution is measured. Not by activity metrics like hours logged or meetings attended, but by outcomes: how many checks were encoded, how much of the constraint surface is covered, whether the same question came back twice. Measurement without action is overhead. Measurement that drives maintenance and improvement is governance.

Sized, for a real portfolio

The numbers are calibrated starting points rather than constants, but they make the shape concrete. A portfolio running eight active Efforts typically needs a guild of ten to fourteen named experts across its domains. At 20 to 30% each, that is roughly three full-time equivalents of expert capacity, plus a coordination rota that costs a fraction of one more, plus the encoding tooling. Against the total cost of eight crews and their fleets, the network is well under a tenth of portfolio spend.

Hold that against what a single miss costs. One policy interpretation discovered at UAT instead of at spec time can consume more capacity in rework than the guild costs in a month, and it is never one.

Treated as volunteer work

The pattern is predictable. In the first quarter, enthusiasm runs high and experts show up. In the second quarter, the home function gets busy and attendance drops. By the third quarter, the network exists on paper and operates on goodwill, which means it operates when convenient.

The failure is not dramatic. It is incremental. Each missed contribution is a piece of context that was not encoded, a check that was not written, a playbook that was not updated. Individually, each gap is tolerable. Collectively, they produce a knowledge surface that no longer tracks reality, and a portfolio that is running on assumptions nobody has validated in months.

The Alternatives That Keep Getting Proposed

The dedicated knowledge team. Extract the experts, staff a standing group, make availability someone's whole job. It fails on the same law the Decision Owner's 40% exists to defeat. Knowledge is refreshed only by doing the business, and a fully extracted expert is a fading one. The guild stays in the business because that is where the knowledge is made.

The documentation drive. Fund a quarter of wiki-writing instead. The result is storage, and storage is not holding. A stored page cannot notice the moment it becomes relevant, and nobody trusts it enough to build against unchecked. Encoding into checks that run on every push is what makes knowledge active rather than archived.

Hiring it. A strong new hire carries another company's domain. What the network runs on, this company's policy history, this customer base, these regulators, exists only inside the building and takes years to acquire. Hiring grows the guild

eventually. It does not substitute for releasing the people who hold the knowledge now.

The one-deep domain deserves naming at funding time. Where a single person holds knowledge the whole portfolio depends on, the duty is encode-first, so every contribution lands in the harness and the graph before it is needed twice, and a second slice of guild time gets funded early in that domain, apprenticed beside the expert while the expert is still in the room.

The Health Test

A funded network must demonstrate returns, and the returns have a specific shape.

Expert hours ↓ per outcome trend down while coverage of domain constraints trends up. That is the compounding signature. If both metrics move in the same direction, something is broken.

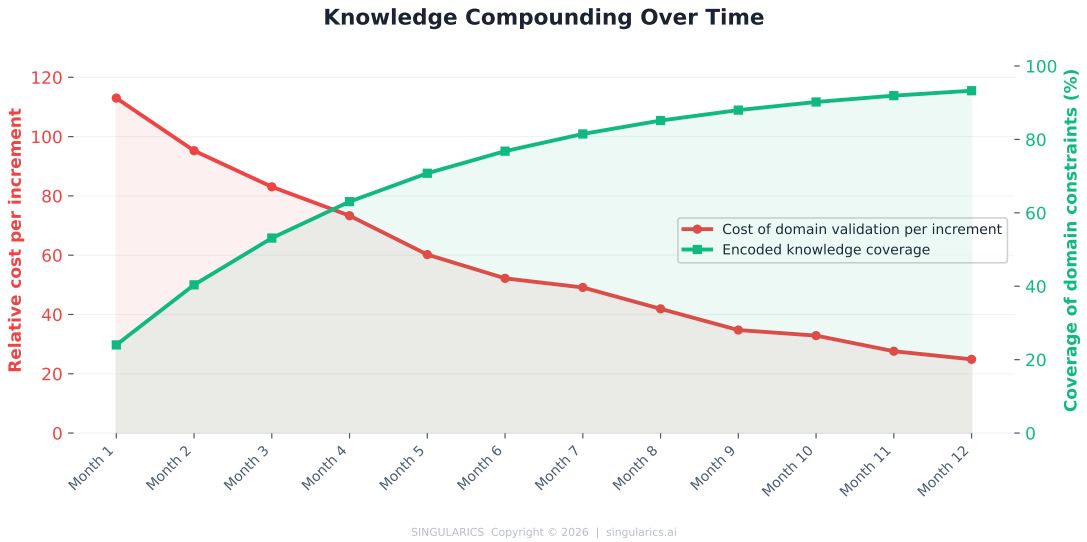

Figure 1: The compounding signature. As encoded knowledge accumulates, the cost of domain validation per increment falls while the coverage of domain constraints rises. This only happens if Encode is funded and the knowledge graph is maintained.

The paired metrics

Expert hours per outcome falling alone could mean experts are being cut. It is only meaningful when coverage is rising simultaneously, because that proves the reduction comes from encoded knowledge doing work that used to require a person in the room.

Coverage rising alone could mean the organization is writing checks nobody maintains. It is only meaningful when expert hours per outcome are falling, because that proves the checks are actually absorbing demand rather than accumulating as dead weight.

The two together are the compounding signature. Each contribution leaves behind checks in the harness and context in the graph that reduce the demand for the next contribution. The network becomes more capable with every cycle, which is what makes a finite guild of part-time experts viable across a growing portfolio.

The watch list

Four indicators, checked monthly:

1. **Expert hours per landed outcome.** Should trend down over quarters. If it trends up, contributions are not being encoded. Concretely: if the third outcome in the same domain needs the same expert hours as the first, the encode step is failing.
2. **Constraint coverage in the harness.** Should trend up. If it plateaus, maintenance has stalled or new policy is arriving faster than it is being encoded. Map coverage against the full constraint surface at quarterly rebalance. Any domain where coverage is below 50% after two outcomes is a red flag.
3. **Repeat question rate.** How often the same question comes back to the network from a different team. A healthy rate is near zero. A rising rate means the encode step is being skipped. Track this by requiring the expert to check the knowledge graph before answering: if the answer is already there, the team missed it; if it is not, the contribution was not encoded.
4. **Stale context incidents.** How often a team works from outdated knowledge graph entries. Any nonzero count is a maintenance signal. A single stale-context incident that reaches production is worth more attention than a month of healthy metrics, because it reveals a maintenance gap that the other metrics cannot see.

How to read the two metrics together

If expert hours are falling and coverage is rising, the network is compounding. Every other combination is a problem.

Expert hours falling and coverage flat means experts are being withdrawn, not leveraged. Coverage is rising and expert hours are flat means the encode step is happening but not reducing demand, which usually means the encoded knowledge is not reaching the teams that need it. Both flat means the network has stalled. Both rising means the portfolio is growing faster than the network can compound, and the guild needs to be larger or the encode step needs to be more aggressive.

Skipping It

The failure mode is not loud. There is no incident, no outage, no missed deadline on the day the investment is skipped. The damage is cumulative and silent.

The network becomes a rota of interruptions

Without funded time, experts participate when they can. Their contributions are reactive rather than planned. They answer the question in front of them, but nobody encodes the answer. Next month, a different team asks the same question. The expert answers it again. The third time, they are frustrated. The fourth time, they stop showing up.

The same questions come back monthly

This is the clearest diagnostic. If the network is fielding the same questions repeatedly, the encode step is not happening. Every repeated answer is a contribution that should have been a check in the harness or an entry in the knowledge graph. Instead it was a conversation, and conversations do not scale.

> *If nothing outlives the conversation, the network is consulting rather than governing, and the same question comes back next month.*

Experts burn out

The worst version of this failure is quiet attrition. Experts who are repeatedly pulled from their primary work to answer questions that should have been encoded will eventually protect their time by becoming unavailable. They do not resign from the network. They simply stop responding promptly. The network degrades one expert at a time, and each departure makes the remaining members' load heavier.

Quality degrades invisibly

Without encoded knowledge, quality depends on whether the right expert happened to be in the room when the spec was written. Some days they were. Some days they were not. The specs that missed a constraint look the same as the ones that caught it until the work ships and a downstream consequence surfaces. By then the team has moved on, the context is cold, and the rework is expensive.

The harness stays green because it only checks what was encoded, and what was not encoded cannot fail. The organization develops false confidence in a harness that covers 60% of the constraint surface and does not know what the other 40% contains.

Knowledge as a Compounding Asset

The distinction between knowledge as a cost and knowledge as an asset is the distinction between a network that decays and one that grows.

A cost is consumed. You spend it and it is gone. If domain expertise is treated as a cost, every contribution is a withdrawal from the expert's time, and the organization's posture toward that withdrawal is to minimize it. Fewer hours, fewer sessions, fewer reviews. The incentive runs against the investment.

> *Every contribution leaves something behind. Checks in the harness, context in the graph. The cost of the next increment falls while confidence rises.*

An asset compounds. Each contribution leaves behind an artifact that does work in the future without further expenditure. A check in the harness runs on every push, for every team, forever. An entry in the knowledge graph informs every spec that touches its domain. A playbook updated after a monthly session prevents every future team from making the mistake that prompted the update.

The compounding mechanics

The mechanics are specific and worth naming.

Encode, not explain. When an expert contributes a constraint, it is written as a check in the harness and as an entry in the knowledge graph. From that moment, the constraint is enforced automatically on every push. The expert's judgment runs continuously without the expert in the room.

Reuse across teams. Because teams are fluid and reform between outcomes, the encoded knowledge does not belong to a team. It belongs to the portfolio. Every new team that forms inherits the full knowledge graph and the full harness. They start with the accumulated judgment of every expert who contributed before them.

Diminishing marginal cost. The first outcome in a domain is expensive in expert time, because the knowledge graph and harness coverage for that domain start near zero. A typical first outcome requires 15 to 20 hours of expert time across its life. The second outcome in the same domain is cheaper, because the checks and context from the first are already there, typically 8 to 12 hours. By the fifth outcome, the expert's contribution is a delta review rather than a full injection, and their time per outcome drops to 3 to 5 hours.

Rising marginal confidence. As coverage grows, the probability that a constraint is missed falls. The Decision Owner accepts with greater confidence. The Flow Council escalates less often. The Intent Council sees fewer surprises at the biweekly check-in. Confidence is not a feeling. It is a function of how much of the constraint surface is covered by checks that run on every push.

The portfolio consequence

A portfolio with a compounding knowledge base can grow its active outcomes without linearly scaling its expert guild. The guild gets more productive with each contribution, not more burdened. That is the return on knowledge investment, and it only materializes if the encode step is funded and the knowledge graph is maintained.

Without compounding, the expert guild scales linearly with the portfolio. Every new outcome needs the same hours as the first. The network becomes a bottleneck at exactly the moment the organization wants to accelerate, and the response is either to hire more experts, which takes months, or to skip the knowledge step, which produces the invisible quality degradation described in the previous section.

Making the Business Case

The business case for knowledge investment runs into a predictable objection: it looks like overhead. Time allocated to network membership is time not spent on the expert's primary function. Monthly sessions are meetings. Knowledge graph maintenance is work that produces no shipped feature.

The objection is understandable and wrong. It is wrong because it compares the cost of the investment to zero rather than to the cost of not making it. The

strongest form of the objection is this: “Our experts are already overloaded. We cannot afford to take 20 to 30% of their week away from their primary function.” The answer is arithmetic. If an expert spends 4 hours per month answering the same unencoded question from different teams, they are already losing 48 hours per year to reactive pulls. The 20 to 30% network commitment is roughly 400 to 600 hours per year, but it reduces the reactive pulls toward zero and leaves encoded checks that run forever. The breakeven typically arrives by the end of the second month.

The math of repeated expert pulls

Consider a single domain constraint, say a policy rule that affects regulated decisions. Without encoding, every team that touches that domain asks the same expert the same question. Each pull costs 30 to 60 minutes of the expert’s time and 2 to 4 hours of the team’s time while they wait.

5 teams × 12 months If five teams each pull the same expert once per month for the same unencoded constraint, that is 60 interruptions per year. Encoding the constraint once costs one hour and a check in the harness. The breakeven is the second pull.

With encoding, the constraint is written as a check in the harness and an entry in the knowledge graph. The first team’s pull costs the same 30 to 60 minutes plus 30 minutes to encode. Every subsequent team inherits the check automatically. The expert is never asked again. The breakeven point is the second team, the second month, or the second outcome, whichever comes first.

For a portfolio with more than one active outcome, which is every portfolio of meaningful size, the return on encoding exceeds the cost within weeks.

The cost of invisible quality degradation

The harder argument to make is the one about quality, because the cost of skipped knowledge is invisible until it is not. A production incident caused by a constraint that should have been in the harness costs more in investigation, remediation, and trust than a year of network funding. An audit finding caused by a policy that was explained but not encoded costs more in regulatory attention than a decade of monthly sessions.

These costs are real but unpredictable in timing, which makes them easy to dismiss in a quarterly planning conversation. The business case should not rely on fear of a specific incident. It should rely on the structural argument: a harness

that covers 60% of the constraint surface will eventually miss something in the other 40%, and the only question is when, not whether.

How to frame it

Frame knowledge investment the way you frame any infrastructure investment. Nobody argues that CI/CD pipelines are overhead, because the alternative, manual deployment, is visibly worse. The same logic applies to the knowledge network. The alternative, ad hoc expert pulls, unencoded constraints, and conversations that do not scale, is visibly worse once you measure it.

Three framing moves that work with executives:

1. **Show the repeat rate.** Count how many times the same question was asked by different teams in the last quarter. Multiply by the cost of each pull. That number is the cost of not encoding.
2. **Show the coverage gap.** Map the constraints that are in the harness against the constraints that exist in policy. The gap is the surface area where quality depends on luck rather than on structure.
3. **Show the trend.** If the network is funded and encoding is happening, expert hours per outcome will be falling and coverage will be rising. Plot them together. The compounding curve is the most compelling argument, because it shows returns that grow rather than costs that accumulate.

The executive who sees knowledge investment as overhead is comparing it to a world where domain expertise arrives for free. That world never existed. The expertise was always paid for. It was paid for in expert time that was not budgeted, in rework that was not attributed, and in quality gaps that were not measured. Making the investment explicit does not create a new cost. It makes an existing cost visible, accountable, and for the first time, compounding.

Failure Modes Specific to Knowledge Investment

Each of these is a failure of funding or structure, not of individual performance.

Network becomes a help desk. Experts are pulled in reactively after teams stall, rather than injected proactively before the build starts. The tell: experts report that they are “always firefighting” and never contribute to the knowledge graph. The fix: restore the push model driven by daily planning (see above) and enforce the rule that every injection session ends with something encoded.

Decision Owner is a proxy. The named owner cannot answer basic domain questions and routes everything to the network. Acceptance decisions take more than a day. The tell: the owner says “let me check with” more often than they

say “yes” or “no.” The fix: replace the owner. Do not coach around this one. A proxy owner is worse than no owner, because no owner sends the outcome back to Next where it waits, while a proxy owner lets the outcome sit in Now and stall silently.

The encode step is skipped under time pressure. The expert answers the question in the room but nobody writes the check or updates the knowledge graph. The tell: the same question comes back from a different team within a month. The fix: make encode the exit criterion for every injection session. The expert does not leave until something is written down.

Home function reclaims expert time. The manager who agreed to release 20 to 30% of the expert’s week quietly reassigns them to home function work. The tell: the expert’s attendance at injection sessions drops below 80%. The fix: treat the release agreement the same way the Decision Owner’s 60% release is treated. It is a capacity commitment, not a handshake. Raise it at weekly calibration and escalate if it persists.

Monthly session becomes a status meeting. The two-hour session fills with updates and presentations rather than encoding work. The tell: the session produces no new checks in the harness and no updated entries in the knowledge graph. The fix: time-box hour one to absorption and hour two to revision, and require encoded artifacts as the output. If the session produces no artifacts, cancel the next one and use the time to figure out why.