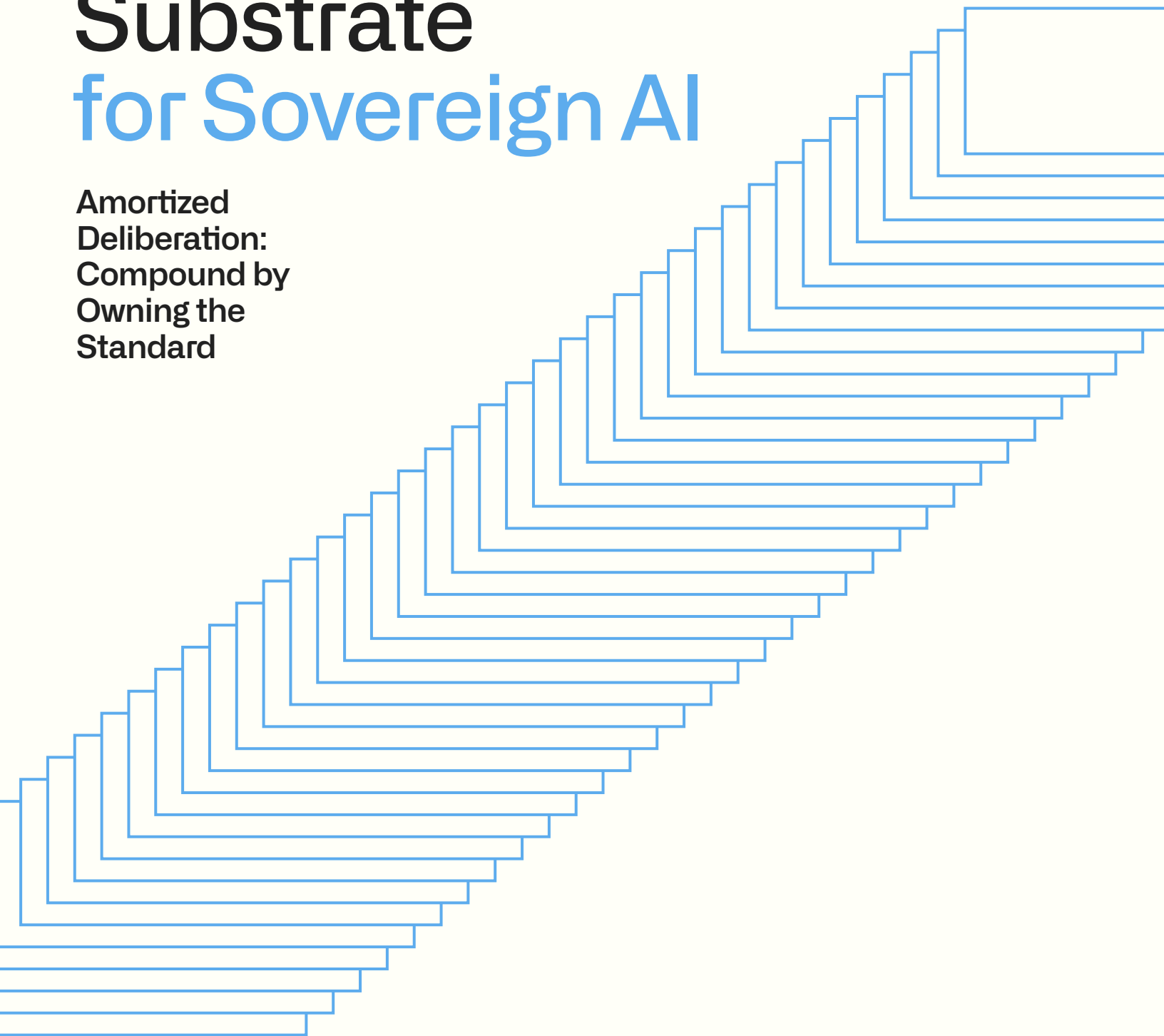
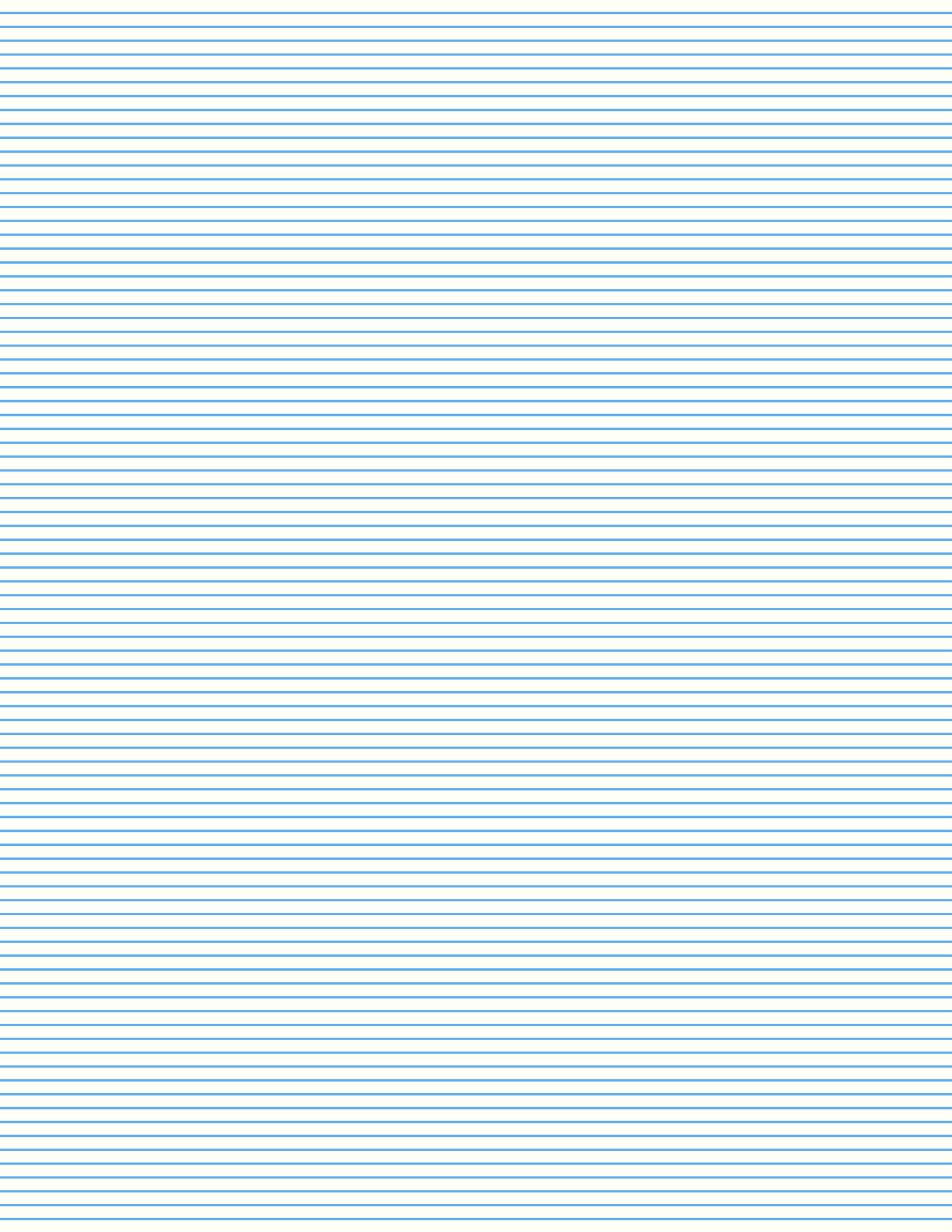


# Evals as the Substrate for Sovereign AI

Amortized  
Deliberation:  
Compound by  
Owning the  
Standard





[ 1 ]

# From Agentic Demos to Reliable Operational Decisions

The most common anti-pattern in the world of “agentic” demos is building systems that ask the model to do the whole job and force the user to clean up the result. The result is a demo that looks technically impressive but is disconnected from operational reality: it asks users to sift through dozens of scenarios when they need help reaching one sound decision quickly. The workflow also leaves value on the table when it fails to draw on the model’s reasoning in decisions where it can make a meaningful difference.

---

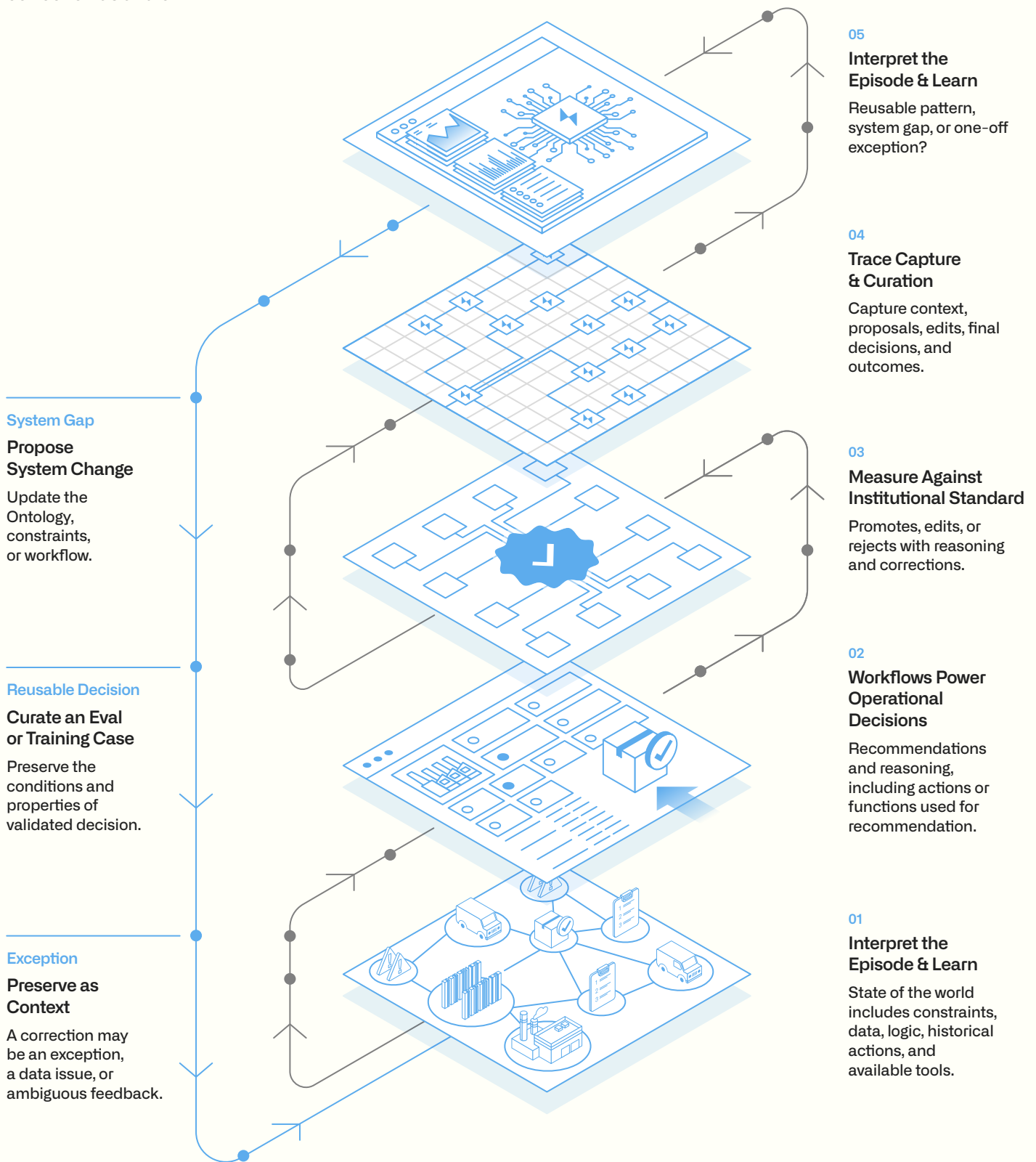
The most effective antidote to demoware is AI built as a suit the operator wears, one that serves the specific decision the operator can make faster and with greater confidence. Does it help a farmer grow more crops per acre? Does it help an insurer set better rates while keeping its risk exposure within limits? Does it help a scheduler produce a feasible plan with less review and rework? Each question shifts the unit of value from the model to the decision.

Real-world use should do more than show whether a system is useful today; it should generate evidence for improving it over time. When a workflow records the information behind a recommendation, the operator’s corrections and reasoning, and what happened as a result, the organization can turn repeated decision-making into tribal knowledge it can reuse. The decision trace allows you to create a compounding Ontology (so you own an increasingly valuable engine of value) and a Fine-Tuning Ontology (training datasets that allow you to fine-tune models so that you can own the means of AI production.)

In this document, we share observations from the field on how to achieve these compounding results, the anti-patterns that prevent institutions from doing so, and the architecture through which operational excellence can become an asset the institution controls.

## Operational Decisions Become Institutional Memory

Decision traces preserve curated context to reuse learnings without treating every correction as a rule.



[ 2 ]

# Beyond Determinism

# We typically see two types of systems: repeatable and adaptive.

---

## 1 / Repeatable Systems

Repeatable systems rely on repeatable rules, heuristics, or processes to achieve the intended enterprise decision. They comprise the majority of workflows.

---

## 2 / Adaptive Systems

Adaptive systems rely on deeper intuitions about the problem. They are difficult to encode or may even conflict with simplistic representations. These novel workflows are less common, but they often represent the true differentiation (or alpha) of the institution.

---

It is tempting to think that these correspond one-to-one to deterministic and non-deterministic processes, but that misses the nuance of the real world. Plenty of processes are repeatable but non-deterministic. Portfolio risk assessments based on simulations can yield different estimates while following an established, *repeatable* procedure. Conversely, a novel scheduling problem may be solved by a deterministic optimizer once its objectives and constraints have been represented correctly.

The useful question is how much of the work is already captured in the institution's Ontology, procedures, tools, and evals. The same workflow can contain both repeatable and adaptive work:

AIP Logic may handle the known path, while AI FDE or AIP Analyst helps the operator investigate the cases that fall outside it.

One reason so many AI demos fail to become useful production systems is that they treat every decision as a new problem for the model to reason through. The system starts from scratch instead of drawing on lessons and constraints learned in the field. Teams often fail to retain what the system proposed, how operators changed it, and what happened next. Without that record, they have little evidence for improving the next decision.

## System Types

### Cont.

In contrast, production workflows built on Foundry rely on a combination of deterministic and non-deterministic components: data integration, Ontology, actions, functions, models, and logics. For a marine aftermarket sales company, for example, Palantir developed a use case to identify the correct part that customers are looking for by piecing together bill of materials, purchase histories, and maintenance logs, which all come together in an auto-quotation system which decreased time to quote to customers from days to minutes. It primarily relies on custom, deterministic tooling built to find the part the customer is looking for. Model reasoning is only used when the case is novel and there are conflicts in responses from deterministic paths.

At this point, it may be tempting to conclude that workflows should be as deterministic as possible and that model intelligence has no role to play. But discovery tasks need a different approach. The right balance depends on the business outcome: some processes call for consistent, rule-bound decisions; others benefit from the model's ability to generate unexpected possibilities. In those cases, the model's output should serve as a source of ideas and not as established fact.

For example, annual healthcare fraud in the United States is estimated to cost more than \$100 billion per year. While known patterns of fraud can be modeled as deterministic rules, unknown patterns of fraud require increased investigation and verification but are difficult to identify. Palantir partnered with a healthcare technology company to create a Claim Fraud Cockpit that uses long-running agents to comb through healthcare claims, pharmacy networks, blacklisted customer and doctor lists, amongst other data sources to identify fraud hypotheses for fraud analysts to confirm and review.

Non-determinism is valuable when the task is discovery: generating testable hypotheses can justify false positives, provided the system clearly separates observed facts from inference. But when the operator needs an executable decision, returning multiple unresolved possibilities merely transfers the work to the user.

Start by understanding the problem before choosing the approach. Decompose the work into what is already covered by a validated procedure and what still requires exploration, then create the combination of procedures, models, and human judgment that fits reality.

A fraud pattern that required open-ended investigation yesterday may become a repeatable check tomorrow.

[ 3 ]

# Amortized Deliberation and Evals as the Substrate for Sovereign AI

## Introduction

The mechanism for making that transition is the eval. An eval is a way to test whether a system makes an acceptable decision under the conditions that matter to the business. For an agent, that means more than checking whether an answer is correct. An eval should capture the situation the system was responding to, the information and tools available to it, the decision it made, and the accuracy of that decision.

Operational decisions often have multiple valid solutions. The question is not whether the system reproduced one historical response exactly, but whether it respected the constraints, preferences, and outcomes that make a decision acceptable.

This is why evals are a source of institutional alpha. Models and model capabilities are also available to your competitors. What remains specific to your institution is your accumulated understanding of what a good decision looks like in its operating context. That understanding is encoded in the cases, standards, and outcomes that make up its eval suite.

## There are two ways this can happen.

---

### 1/ By Hand, into Code

An operator exercises judgment. An engineer then generalizes the lesson in a second round of human reasoning and encodes it as a rule, constraint, tool, or procedure. This path is discrete, explicit and auditable: the institution can inspect exactly what was written down, argue with it, version it, and change it.

---

### 2/ Distilled, into Weights

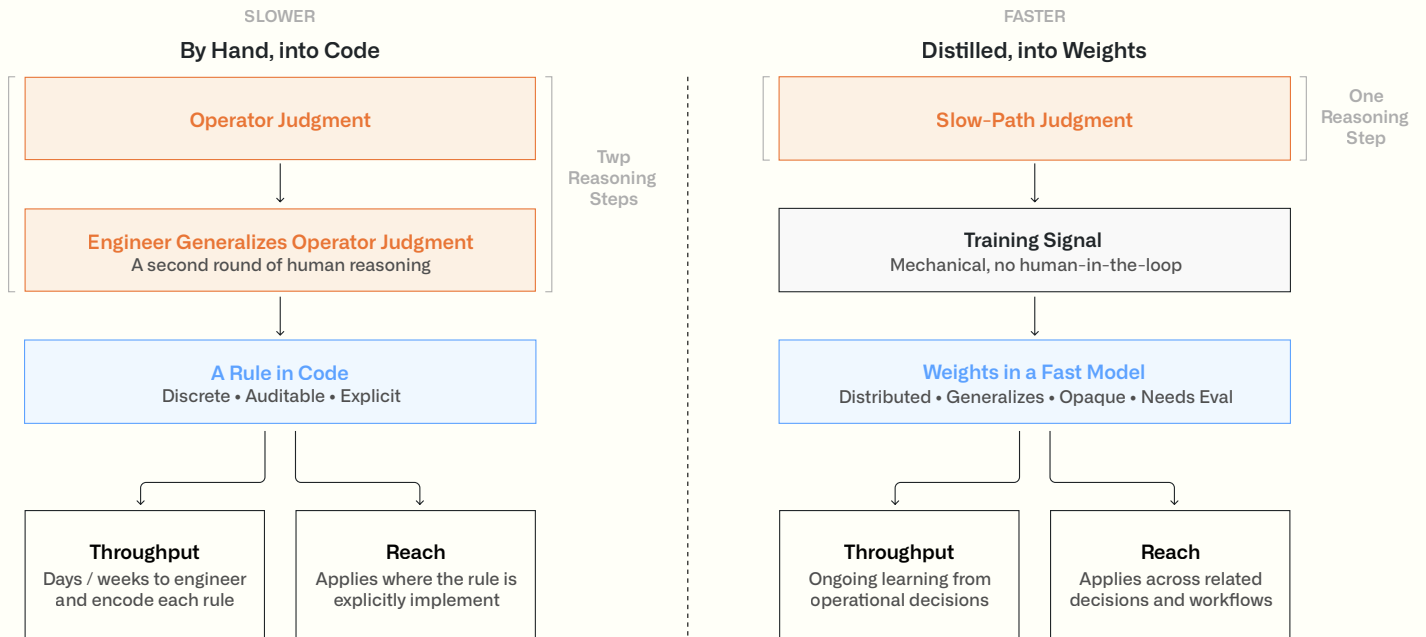
Post-training can absorb repeated examples, operator preferences, and validated decision patterns into a model's learned behavior. This gives the institution a way to scale its operational knowledge across many related situations rather than encoding every variation as a separate rule. It is therefore an important part of sovereign AI, particularly when the knowledge is broad, difficult to express explicitly, or useful across many workflows.

---

The tradeoff is that this knowledge is distributed through the model rather than represented as a single inspectable rule. It can generalize beyond the examples used during training, but it is harder to inspect and modify precisely. Evals, Ontology constraints, tools, and procedural checks remain important because they define the standards against which the post-trained model is tested and they help bound its behavior in production.

The end goal is not a complete rulebook. It is an amortized policy: a system in which the institution no longer pays the full cost of reasoning through decisions it has already understood, while preserving room for judgment where the problem remains genuinely novel.

## Instinct is Expensive Reasoning, Amortized



The illustration above shows what happens to that understanding over time. Both columns turn orange into blue. Both are ways of converting expensive deliberation into a faster policy. The difference is where the policy lives, how inspectable it is, how broadly it generalizes, and how it must be maintained.

Instinct is expensive reasoning, amortized. A person or model reasons through a difficult case in the orange space. If the lesson is validated, it can move into the blue space: already decided and written down.

### 3.1

## Why You Need Evals Before Agent Deployment

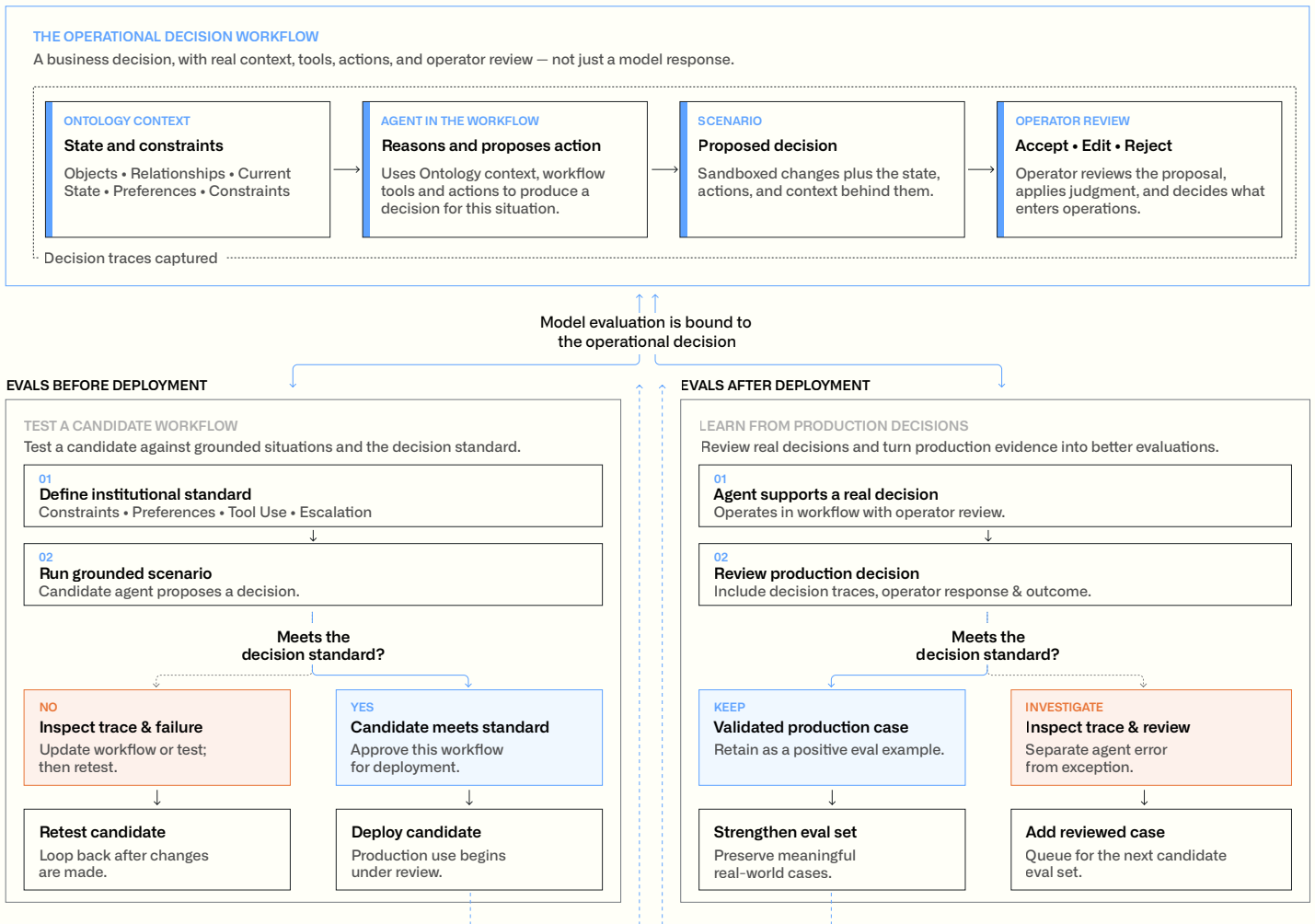
For most operational processes, the decision space is wide enough that it is impossible to encode every edge case before deployment, but structured enough that parts of it can become code, constraints, tools, reusable policy, or training signals. The challenge is to determine which parts should move into which layer.

The usual framing is that an eval is built after the agent: assemble a test set, define a rubric, run the model, and measure the answer. That treats evaluation as separate from the workflow that should continually improve it.

Initial evals are necessary before deployment. But production decisions should generate the evidence that extends and corrects them. In an operational workflow, the evidence for an eval should be captured at the same moment as the decision.

### Evaluate the Model within the Operational Workflow

Production decisions should generate the evidence that corrects evals.



# The eval suite should test the properties that make a decision acceptable, including:

- 
- 1 / Satisfying hard constraints
  - 2 / Respecting operational preferences when objectives conflict
  - 3 / Responding correctly to changed inputs
  - 4 / Using the appropriate data, tools, actions, and policies
  - 5 / Minimizing unnecessary disruption
  - 6 / Providing a traceable explanation
  - 7 / Escalating genuinely novel cases
- 

The suite should combine several sources of evidence: initial human-authored cases, targeted tests for situations production has not yet exposed, and cases derived from real operational decisions. Rather than trying to reproduce an earlier model response exactly, the goal is to evaluate a decision against the relevant conditions, standards, and outcomes.

## 3.2 Generating Evals through Production Usage

A Scenario is one concrete way to generate this kind of evidence. It provides a separate “sandbox” environment where an agent can propose changes and an operator can inspect them before accepting the changes. It can preserve the state of the world, the objects read, the actions and functions called, the proposal, and the operator’s changes before accepting, rejecting, or merging the result.

This is what it means to think of Scenarios as an engine for eval creation: the broader asset is the eval, while the Scenario is one product mechanism for capturing a richly labelled decision process.

HD Hyundai Oilbank’s Production Scheduling System (PSS) is an example that uses Scenarios as evals. PSS is a scheduling and optimization workflow for gasoline blending at an oil refinery. The blender must (i) combine intermediate products to produce gasoline of a certain chemical specification, (ii) meet demand manifested in shipments and ship schedules, and (iii) ensure that intermediate-product tanks do not violate their upper and lower inventory limits.

PSS is useful because it contains both a deterministic core and a decision space where operational knowledge remains incomplete. Its scheduler, optimization code, Ontology reads, external scheduling integrations, and Scenario mechanics can be deterministic. The model is useful where the operational problem is not fully written down: it can surface a proposal, identify a conflict, or expose a consideration that the existing procedure does not yet represent.

In the workflow, the agent sees the current inventories, shipment schedules, production constraints, tank limits, quality specifications, and other relevant Ontology state. It proposes a schedule inside a Scenario rather than writing directly to production. An operator then inspects the proposal and may merge it unchanged, edit blend quantities, change the order of operations, reject it entirely, or delete a proposed action.

## Each outcome contains information:

- A merge says that the proposal was acceptable for that state of the world
- An edit says that the system found a plausible path but missed a preference, constraint, or piece of context
- A rejection says that the proposal was not merely suboptimal but unusable

These signals are valuable, but their meaning depends on context. A merge is not proof of correctness. An edit may reveal a missing constraint, a temporary exception, a data-quality issue, or a personal preference. A rejection may reflect a changed shipment schedule rather than an unusable proposal. The feedback is more useful because it is attached to the full decision context, not just recorded as an isolated thumbs-up or thumbs-down.

## The Scenario is therefore a richly instrumented source of eval cases. It can preserve the:

- State of the world the system was responding to
- Decision space and constraints available to it
- Tools, actions, and functions it used
- Proposal and its reasoning or explanation
- Operator's final decision
- Delta between the proposal and the accepted result

The delta is especially valuable. An operator's edit is not only a correction to the current schedule. It is evidence about the boundary between what the system already knows and what still requires human deliberation.

### 3.3 Continuous Improvement of the Institutional Standard in Practice

At the beginning, the agent and operator jointly occupy a large exploration space. The model handles what is novel or not yet represented; the operator supplies judgment where the Ontology and procedural layer (system's explicit rules, tools, and workflow steps) are incomplete. The workflow preserves evidence so the same deliberation is not paid for — in time and tokens — next time.

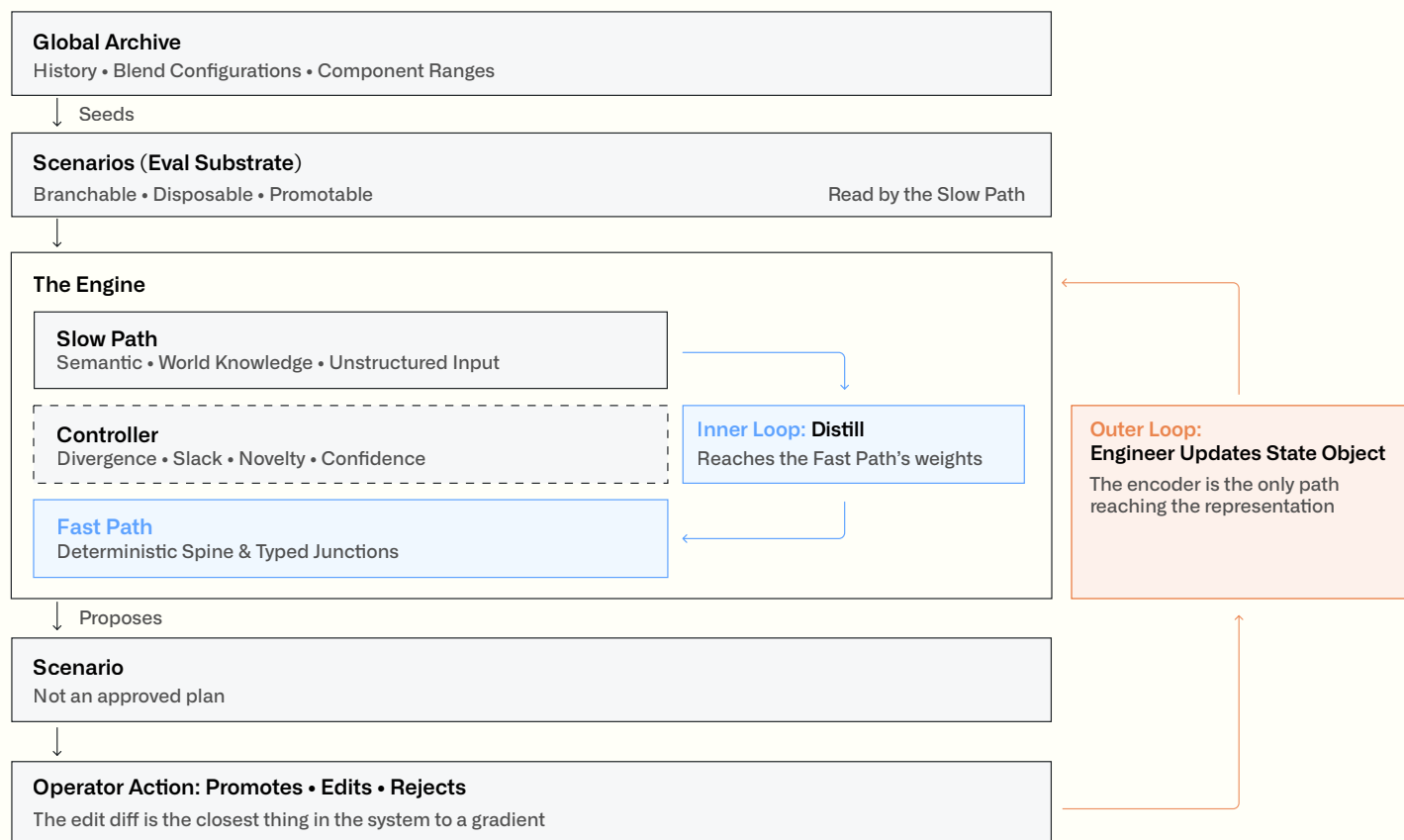
## The loop is:

- 1 / The system proposes or supports a decision
- 2 / The operator accepts, edits, or rejects the result
- 3 / The workflow records the relevant state, proposal, actions, and feedback
- 4 / The team determines whether the episode reveals a reusable requirement, defect, or one-off exception
- 5 / A proposed improvement is tested on separate cases and approved
- 6 / Subsequent runs use the improved representation, tool, or procedure, while evals check that the improvement holds over time

A historical Scenario can supply the input state and evidence about the expected properties of a good result. An accepted schedule can serve as a reference point. And an edited Scenario may be even more informative because the delta identifies where to investigate a possible gap in the system's policy.

## Two Nested Loops, Running at Different Speeds

The inner loop makes the system better at what it can already see. Only the outer loop lets it see something new.



But not every correction should become a rule. A similar edit under a Scenario might reflect temporary maintenance, a late laboratory result, or an operator preference rather than a general requirement. A correction may call for a data quality fix, a better tool, a new constraint, or no generalization at all. Cases used to develop a change cannot be its only proof of generalization, and changes to the institutional bar of what good looks like should be approved, versioned, and reversible.

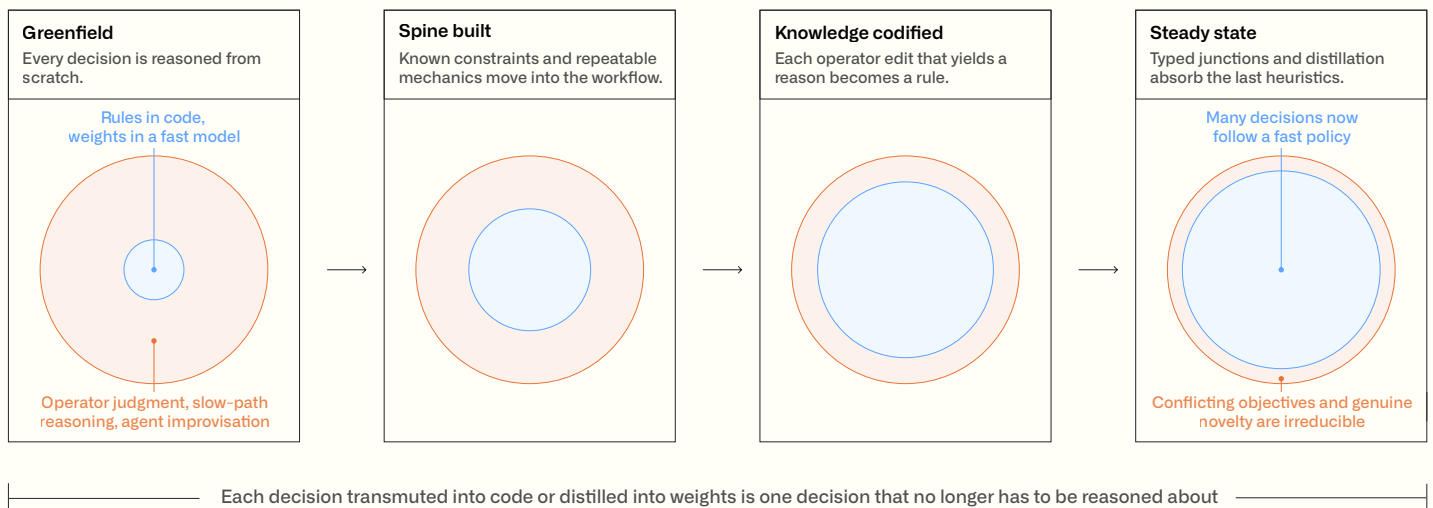
## 3.4 Beyond Scenarios

The illustration below captures the relationship between evals and amortized deliberation. The outer circle — the whole business problem — does not get smaller. What changes is how much of that problem still has to be reasoned through on every run.

In the greenfield state, nearly every decision is made from scratch. The system has little procedural memory, so the operator and the model occupy most of the decision surface.

### A Shrinking Decision Surface Is the Goal, Not a Side Effect

The outer circle represents the problem and remains the same size, but what changes is how much of it still has to be thought about.



## Let's take the evolution of the PSS use case as an example.

---

As the spine was built, obvious constraints and repeatable mechanics moved into code and workflow: settling times, minimum shipment quantities, inventory limits, tool preconditions, or known data transformations. The system still had a large space to reason through, but it no longer has to rediscover the basic structure of the problem each time.

As knowledge is codified, operator edits that repeatedly revealed the same underlying reason can become explicit rules, Ontology constraints, AIP Logic functions, better tools, or agent-workflow instructions. Each validated lesson moved one more decision inward — from something that had to be deliberated and confirmed every time into something the system can apply directly.

In steady state, typed junctions, procedural logic, and — where appropriate — distillation into model weights absorb the remaining recurring heuristics. The decision surface has shrunk, even though the underlying business problem has not. The institution is not solving less of the problem; it is paying less to solve the parts it has already understood.

The broader lesson is that amortized deliberation is not the same as maximizing determinism. It is the movement of validated knowledge from expensive, open-ended reasoning into representations the institution can inspect, test, govern, and reuse. In PSS, a Scenario provides one practical path for capturing that knowledge. The eval suite determines whether the lesson holds; the procedural layer — or, where appropriate, the model's learned policy — makes the validated lesson faster and more repeatable.

[ 4 ]

# Sovereignty Means Owning Your Decision Standards

Over time, the process of capturing operational decisions, testing proposed lessons through evals, and encoding validated knowledge into procedures produces an asset distinct from any particular model: a versioned record of what the institution considers a good decision, and the procedures through which that standard is applied.

---

This is the connection between amortized deliberation and sovereignty.

Similar businesses can share substantial parts of their operational structure. Two refiners have essentially the same scheduling problem: tanks, berths, settling times, alarm limits, vessel windows, and chemical constraints. The physics is shared, and much of the equipment is shared. Neither the model nor the data schema is the differentiator, because a competitor can buy both.

What differs is the encoded operational knowledge: which constraints actually bind, which preferences matter when objectives conflict, which situations are exceptions, and why. That is what your people know, and what your competitor's people know differently. This is the company's alpha — it's the reason why your competitors cannot simply copy everything you do and expect the same returns.

Sovereignty means owning the expression of that knowledge: in the Ontology, in the evals that define "good," and in the procedural layer that has absorbed it. The Scenario records the episode, the eval identifies the standard, and the procedural layer makes the standard fast and repeatable.

# Sovereignty means control over the institution's decision standards, and is not limited to access to a model.

---

This gives a cleaner reason why post-training is a later lever rather than the first. Post-trained weights are not the most portable form of your alpha and it shouldn't be the first go-to option. A constraint codified in your Ontology survives a model swap, a vendor outage, and a generational jump that changes how the model reasons.

## The sovereign assets are therefore:

---

- |                     |                                                                                                                                                                      |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1/ Ontology         | The entities, relationships, state, constraints, and actions through which the business is represented                                                               |
| 2/ Evals            | The accumulated examples and human validations that define what a good decision looks like under real conditions                                                     |
| 3/ Procedural Layer | The Ontology with data, logic, action, and governance along with rules, tools, schedulers, policies, and workflow steps that make validated knowledge cheap to apply |
| 4/ Weights          | Useful when they improve performance, but not the most durable place to store business-specific advantage                                                            |
- 

Ownership must be practical, not merely nominal. The institution needs the ability to inspect, export, modify, govern, and use these assets as models and infrastructure change. Otherwise, dependence on one provider may simply become dependence on another.

Sovereign AI work and good agentic architecture are closely related because both require deciding what the business knows, validating that knowledge, and encoding it in a form that the business controls.

## Concluding Thoughts

A standard Foundry workflow (data integration, Ontology, logic, functions, actions) should come first to create the substrate that makes agents useful. The Ontology and actions must already be stood up, deployed, and in use. Operators must already be doing the work. There must be a production baseline, a governed action surface, and a body of decisions from which to learn.

Then the agentic layer can be introduced where it has leverage. It can identify when a new Scenario is needed, gather the relevant state, propose a change, explain its reasoning, and route the proposal to an operator. The operator gets a safe, separate environment to validate the work, and the organization gets a durable record of what happened — and how to improve the system the next time it runs.

Beware of demos that are a palliative measure which no operator can or does depend on in the field. The reliable product that delivers a return on your investment is the workflow that helps a user reach a sound decision, captures the user's decision exhaust, and turns corrections into better system behavior.

