PROFORMA GLOBAL RESEARCH

Governance as Middleware

Matt Rollings, Founder and Principal, Proforma Global

Whitepaper · 2026-05-18

Executive Summary

Governance over an agentic process operating against a financial system must be systematic and deterministic in execution. The pattern that has emerged across the agentic data architecture market in 2025 and 2026 places governance rules in the semantic layer alongside metric definitions and business logic, surfaces them to the agent as context, and trusts the reasoning layer to enforce them. The pattern is broken at the categorical level. Reasoning is non-deterministic by definition. A probabilistic reasoner cannot enforce a deterministic rule, and the cases where it fails are the cases that matter most in any financial system.

The architectural correction is to express governance as deterministic functions invoked at every meaningful step in the semantic traversal a financial agent performs. As the agent navigates from a user's input toward an operation, the multi-dimensional relationships among semantic layers are what the traversal moves through, and the traversal itself invokes the governance functions configured against each step. Each function evaluates against live process state and, where the rule requires it, against live data drawn from the financial system. The function produces a deterministic verdict that determines what the agent encounters next. Navigation belongs to the reasoning layer; the governance functions hold the authority to decide what the agent is permitted to do at each step. This paper sets out why informational governance fails, what the middleware position is, why the architecture produces compliance certifications the reasoning layer cannot, and why the patches the vendor space will reach for do not address the underlying problem.

---

1. The Claim

Governance must be expressed as deterministic functions invoked at every meaningful step in the semantic traversal a financial agent performs. The reasoning layer navigates and proposes; the governance functions evaluate each step deterministically and produce verdicts that determine what the agent encounters next. The current pattern in agentic data architecture stops at informational governance: it places governance rules in the semantic layer, surfaces them to the agent as context, and lets the reasoning layer enforce them through its own judgment. The pattern is widespread, vendor-promoted, and architecturally wrong for any system where governance failures are non-tolerable.

The argument is categorical. Governance in finance is the codified intent that controls be applied to every transaction every time. The class of rules that governance enforces is deterministic by design: a rule produces a binary verdict against a precise condition, and the verdict does not tolerate anything less than 100% certainty of execution. A period is open or it is closed. The same shape extends across the governance surface in finance, which is what gives the rules their structural character. These policies bind the architecture. The architecture must obey them as written, every time, and has no discretionary surface against them.

LLM reasoning produces probabilistic outputs. The same prompt with the same context can produce different outputs, and the probability of compliance can be raised arbitrarily high through engineering effort but will never reach 100%. A reasoning layer that complies with governance 99% of the time is wrong 1% of the time. In finance, the 1% is where the audit committee, the regulator, and the CFO live. Activities such as adjustments to final financial data cannot be permitted without explicit human approval and an audit trail demonstrating process compliance. The disallowance has to be absolute. Statistical likelihood of compliance is insufficient for the class of decisions involved.

The architecture that survives this contradiction separates concerns at the points where the contradiction manifests. The reasoning layer can remain probabilistic, flexible, and exploratory. These are the properties that make it useful for the open-ended work agents are deployed to do. The governance functions have a different mandate. Their properties are determinism, exhaustive coverage of the surface they govern, and refusability. The two mandates cannot coexist in a single layer, and the architectural correction is to give each its own position in the architecture. Middleware is the right layer, as it can intercept, audit, and preempt behavior before the actions can ever be allowed to occur.

---

2. Why Informational Governance Fails

The pattern is straightforward to describe. Governance rules sit in the semantic layer alongside metric definitions, hierarchies, and business logic. The agent retrieves these rules as context whenever it formulates a response or proposes an operation, and the reasoning layer is expected to produce an output that complies with them. The promise sold to the buyer is that the agent, given the right semantic foundation, will respect governance because it has the rules in its working memory. The promise misunderstands what a prompt is. Any input provided to an LLM, including governance rules surfaced as context, is structurally a hint. Hints can be ignored, and they regularly are, under conditions the engineer cannot anticipate. The behavior is

documented across every serious deployment of LLM agents: models violating explicit, unambiguous rules they had read in the system prompt and could quote back on request. These violations are structural properties of the layer. Prompt engineering cannot close them.

The pattern is intuitive because it mirrors how a competent human analyst operates. An analyst reads policy, applies judgment, and produces a compliant action. The architectural mistake is treating the LLM as if it occupied the analyst's position. Human compliance is bounded by accountability and remediation cycles the LLM does not participate in. The model produces the most probable next token given its conditioning. The conditioning is engineered. Consequences do not motivate the model the way they motivate a human analyst.

Consider a single canonical case. A user instructs the agent to post a journal entry for a prior period. The semantic layer holds the period status: closed. The agent retrieves this and reasons about the request, weighing the user's instruction against the period status. In the cases the architecture's designers anticipated, the agent refuses to post and explains why. The designers did not anticipate every case; in the remaining ones, the agent finds a way to comply with the user. The reasoning that leads to compliance can be plausible. The user framed the request as an adjustment to a known reconciliation item, invoked an authority the agent had been told to respect, and continued a pattern of interaction the model's conditioning had reinforced. The context window contained higher-priority information that drew attention away from the period status. The post happens. The audit log records the post. The architecture did not prevent it because no part of the architecture is positioned to.

The failure mode is silent governance violation, typically unaudited and unapproved. The architecture produces an outcome that looks like a normal posting because the agent reasoned its way to it, and the audit trail records the operation as if it had been authorized. The only way to detect the violation is for a human to discover later that the period was closed and the entry should not have been permitted. Reconciliation can catch it, controller review can surface it, an auditor's sampling can find it years later. These are humans catching what the architecture failed to enforce. The architecture itself catches nothing.

The audit problem compounds the failure. When governance enforcement lives in the reasoning layer, the architecture cannot distinguish between an agent that considered the rules and chose to comply and an agent that ignored the rules and happened to produce a compliant output. The two are indistinguishable at the output layer. An auditor reviewing the system can certify the outputs in the sample examined were compliant. The auditor cannot certify the controls were applied, because the controls are buried in the reasoning that produced the outputs, and the reasoning is not deterministic, not reproducible, and not inspectable in the way controls require.

This is the failure mode financial systems specifically cannot tolerate. The cost of probabilistic enforcement scales with the cost of the failures it lets through. In most operational domains the cost is bounded and the failures are remediable through normal customer recovery. In finance a single allowed failure can be irreversible at the moment it occurs and impossible to remediate by anything short of restatement and disclosure. The asymmetry of failure cost is what makes the architectural choice unavoidable.

---

3. The Middleware Position

The architectural correction must express governance as deterministic functions configured against the semantic surface itself, invoked as the agent's request is processed across the multi-dimensional relationships the semantic layers describe. A single shim between agent and execution layer is too narrow a position to cover the surface governance must reach. The traversal does the invoking. The functions do the evaluating. The verdicts produced determine what the next step in the workflow looks like, and whether that step is permitted to happen at all.

The mechanism is the following. As the agent resolves a user's input, navigates the relationships among semantic layers, and assembles the operation a request implies, the architecture continuously evaluates which governance functions apply at each step. The functions are declaratively configured. Each function knows the traversal points it attaches to, the conditions it evaluates, and the deterministic verdicts it produces. The agent does not invoke them. The traversal does, at the points the configuration declares them present. The architecture resembles a traditional rule engine in shape, with one critical difference: it fires on semantic traversal events. The traversal carries the context governance evaluation requires, which data-event triggers do not.

A governance function evaluating a journal entry traversal reads whatever live data its configured logic requires from the financial system, applies the logic, and returns a verdict that shapes the workflow continuation: permitted, redirected into an approval workflow when a configured threshold is crossed, or refused outright when a foundational condition fails. A redirected workflow's first step may itself be a governance function whose verdict determines whether that step proceeds. The architecture composes the cumulative shape the operation experiences through this sequence of deterministic decisions, none of which the agent makes.

A concrete walkthrough makes the mechanism visible. The agent constructs a journal entry against a target period. The traversal invokes a period-status governance function. The function reads the period's current state from the financial system in real time. Open: the workflow continues. Closed without an active reopen workflow: the operation is refused outright. Closed with an active reopen workflow whose approval threshold the entry triggers: the proposed operation is redirected into the approval workflow as a draft, where another governance function evaluates whether the requesting user holds the authority the reopen workflow requires. None of these verdicts depends on what the reasoning layer concluded about the user's intent. The traversal invokes them. The functions evaluate them. The workflow shape changes accordingly.

The intersection of process and governance is what the architecture composes here. Each function applies configured policy against the current process state, reads live financial data where the policy requires it, and produces its verdict at the traversal step where the configuration positions it. The agent must operate within a deterministically governed workflow where it is not afforded decision rights over its own actions, regardless of its reasoning capabilities.

Oracle EPM Cloud, as a representative enterprise financial platform, illustrates the surface the governance functions operate against. A close cycle running across forms, reports, consolidation, and reporting carries state-conditioned controls at every level of its semantic structure, each one the operational form of a governance policy the organization is required to enforce. The middleware position is the architecture that attaches deterministic governance functions to these gating points and invokes them as the traversal passes through, regardless of what the reasoning layer has been told or how it has been prompted.

The audit position changes accordingly, and the change is more substantive than an improvement in record-keeping. The informational governance pattern produces attestations: the agent reports that it considered the rules and chose to comply. The middleware pattern produces certifications: the architecture demonstrates that the controls were applied. The distinction is categorical. Attestation states a belief about what happened. Certification proves what happened, by recording the function invoked, the inputs the function read, the verdict the function produced, and the traversal step at which the function fired. Audit and regulatory frameworks in finance require certification. Attestation does not satisfy them. An architecture that produces only attestations will fail any audit that examines whether the controls were applied, as distinct from whether the outputs the controls were meant to govern happened to look compliant in the sample examined.

4. Why the Available Patches Do Not Address the Problem

The vendor response to this argument will invoke improvements to the reasoning layer, each positioned as the missing piece that will close the gap between probabilistic reasoning and deterministic enforcement. None of them closes the gap because the gap is categorical. Every improvement raises the probability that the reasoning layer produces a compliant output, and compliance probability is bounded above by some value strictly less than one for any conditioning regime. The failures get rarer; they do not stop. Fine-tuning compounds the problem by teaching the model to produce its failures more confidently. The detection burden grows. The failure frequency at the categorical level holds. The architectural problem this paper indicts is a determinism problem. Probability is the wrong frame for it, and engineering effort that raises probability cannot produce determinism. The enforcement has to leave the reasoning layer and live in the governance functions the traversal invokes.

5. Boundary

This paper has argued that governance must be expressed as deterministic functions invoked during the semantic traversal a financial agent performs, that informational governance located in the reasoning layer fails categorically, and that the patches available to improve reasoning-layer compliance do not address the underlying determinism problem.

The paper has not argued against informational semantic layers for what they do well. Definitions, hierarchies, metrics, and business logic legitimately occupy the semantic layer, and the agent's reasoning legitimately consults them. The position is narrower: governance enforcement is a separate concern that must be invoked deterministically wherever the traversal reaches a gating point.

The paper has not prescribed how the governance functions are built. The engineering choices around state representation, policy encoding, and integration with the financial system vary by organization and by the systems being operated against. The position is upstream of these choices. The implementation discipline is downstream, and is the substantive engineering work the position implies.

The paper has not claimed financial systems are alone in needing this. Any domain where governance is deterministic and the cost of silent violation is high has the same architectural requirement. Finance is the case study because the audience is financial and the governance rules in finance are unusually explicit. The architectural claim generalizes.

The paper has not addressed how regulatory governance separates from internal governance. The evidentiary standards regulators demand are categorically different from those internal governance produces. The deployment should structure regulatory governance, such as the controls operating under SOX, in firewalled layers, because bleeding the layers together compromises both. That separation discipline warrants its own treatment.

The diagnostic the paper offers is direct. Where in the architecture does refusal happen, and what evidence does the architecture produce when controls are applied. When refusal lives in the reasoning layer, the architecture is informationally governed; failures are silent and the evidence is attestation that does not certify what happened. Deterministic governance functions configured against the semantic traversal produce enforcement-governed refusal that is loud and audible when it fails, and they produce certifications that demonstrate the controls were applied. Configurations that produce defensible output in regulated financial environments built both the gates and the certifications. The ones that produce confidently wrong output trusted reasoning to substitute for both, and the substitution does not hold.
