WHITE PAPER

The Librarian and the Jury

Two Architectural Patterns for Trustworthy Agentic AI in Regulated Financial Services

Smit Jani, CAITL

AI Governance & Data Strategy Consultant

July 2026

Contents

Executive Summary

Enterprise AI programs are moving from single-model chat assistants to agentic systems: networks of AI components that plan, retrieve information, call tools, and produce work products with reduced human involvement per step. For technology leaders in regulated industries like banking, insurance, and wealth management, this shift creates a governance problem that model selection alone cannot solve. The reliability of an agentic system is determined less by which model you buy and more by the architecture that surrounds it.

This paper examines two architectural patterns that have emerged from production deployments as pragmatic answers to the two most consequential failure modes in agentic systems:

Neither pattern is exotic. Both mirror controls that regulated institutions already understand: records management on the input side, maker–checker segregation of duties on the output side. That familiarity is precisely their value. They give risk, audit, and technology teams a shared vocabulary for building agentic systems that can withstand model risk scrutiny, and they can be adopted incrementally, starting with a single routing layer or a single independent critic.

The paper describes each pattern, explains how they compose into a full input-to-output control chain, maps them to governance frameworks familiar to financial services leaders, and closes with a phased adoption roadmap.

1. The Problem: Capability Has Outrun Control

The dominant conversation in enterprise AI during 2024–2026 has been about capability: larger context windows, better reasoning, tool use, multi-agent orchestration. The quieter and more consequential conversation is about control. When an agentic system drafts a client communication, reconciles a data discrepancy, or proposes a remediation for an audit finding, two questions decide whether the output can be trusted:

Most early agentic deployments answer neither question well. On the input side, teams discovered that bigger context windows invite context pollution: loading every plausibly relevant document, tool definition, and conversation fragment can degrade reasoning quality rather than improve it, and consumes the working memory the model needs to think. Controlled studies of long-context models point the same way, finding that a model uses information reliably when it sits at the beginning or end of a long context but much less reliably when it is buried in the middle. On the output side, teams discovered that asking a

model to review its own work tends to produce flattering reviews, a self-preference effect that research on LLM evaluators has documented directly. A single model acting as author, judge, and jury is the AI equivalent of a trader approving their own trades, a control failure any bank examiner would flag on sight. The research adds some nuance here. Self-evaluation performance varies quite a bit with task type, prompting strategy, and model architecture, and it can be reasonably effective for narrow factual checks while falling off sharply for creative and open-ended work. But that variability is really the governance problem: a check whose reliability swings with the task is not one an examiner can rely on. The trader analogy is imperfect, of course. A model has no motive, and its self-reviews fail through correlated blind spots rather than self-interest. The structural conclusion still holds, though: trustworthy evaluation needs an evaluator whose errors are independent of the author's.

These are not hypothetical concerns. A Canadian tribunal held Air Canada liable in 2024 after its customer-service chatbot confidently supplied an incorrect bereavement-fare policy, and a US federal court sanctioned lawyers in 2023 (Mata v. Avianca) for filing a brief full of citations an LLM had fabricated. In both cases there was no authoritative input layer governing what the system relied on, and no independent review standing between the output and the customer or the court. Comprehensive failure-rate data for agentic systems is still scarce, which is itself a symptom of the instrumentation gap the roadmap in Section 6 begins with, so institutions should treat their own Phase 0 logging as the primary source of incident evidence.

The Librarian and Jury patterns address these two failure modes directly. They are worth examining together because they bracket the agentic pipeline: one governs everything entering the system, the other governs everything leaving it.

2. The Librarian Pattern: Governing the Input Side

2.1 What it is

In a traditional retrieval-augmented generation (RAG) setup, a similarity search pulls document chunks into the prompt and hopes for the best. The Librarian pattern replaces that hope with judgment. A dedicated curating layer, which may itself be an agent, takes responsibility for the information supply chain: deciding which knowledge source to consult for a given question, what to load into the working context, what to withhold, and when to flag that information may be outdated.

A useful mental model is the difference between a warehouse and a research library. The warehouse hands you every box that matches a keyword. The librarian asks what you are

actually trying to do, knows which collection is authoritative for that question, brings you the three documents you need, and tells you when the edition on the shelf has been superseded.

An obvious objection follows: if the librarian may itself be an agent, then the librarian can also fail. It can misroute a query, over-trust a stale source, or hallucinate a retrieval decision outright. The pattern does not eliminate that risk, but it does relocate and shrink it. A curating layer has a narrower and more testable job than an open-ended reasoning agent, its decisions can be constrained by deterministic guardrails like allow-listed sources and mandatory staleness checks, and because every routing decision is logged, its failures are visible and sampleable rather than silent. Institutions should treat the librarian as a model risk component in its own right, one that is monitored, periodically validated, and subject to the same review discipline as the agents it feeds.

2.2 How it shows up in practice

Two production variants illustrate the range of the pattern.

Tiered capability loading. In agentic development platforms, one documented implementation keeps only a small set of high-frequency skills registered directly with the model, while hundreds of specialized capability documents remain invisible until explicitly loaded for the task at hand. The design exists because eagerly loading every tool and skill definition can consume a meaningful fraction of the context window before the agent has read a single line of the actual task. The librarian layer preserves working memory for reasoning by loading capability knowledge just-in-time.

Knowledge-base routing. In enterprise knowledge applications, the "agent-as-librarian" variant places an LLM in charge of the retrieval decision itself: which system to query (documentation, code repositories, ticketing history, chat archives), how to combine results, and when to flag staleness. Practitioners who have deployed this report a counterintuitive lesson from their own experience: the routing decision can matter more than the retrieval quality of any single source, and the hardest engineering problem is less retrieval than teaching the agent to say "I don't know" or "this may be outdated" rather than assembling a confident answer from weak evidence. There are practical techniques for exactly this, including uncertainty estimation over retrieval results, explicit confidence thresholds below which the agent has to qualify or decline to answer, and refusal training that rewards calibrated abstention over fluent guessing.

A portability caveat applies to both variants. How a librarian layer gets implemented depends heavily on the underlying agentic framework, including its context-management model, tool-registration mechanics, and orchestration hooks. The pattern's logic generalizes

but specific implementations often do not, so teams should expect meaningful re-engineering when moving between frameworks or deployment contexts.

2.3 Why governance teams should care

For a regulated institution, the Librarian pattern is not merely a performance optimization. It is the architectural expression of data lineage and records-management discipline applied to AI. When a curating layer mediates every retrieval, the institution gains three things auditors ask for:

Without a librarian layer, an agentic system's effective "data governance" is whatever the similarity search happened to surface. No model risk framework should be comfortable with that. The discipline also has a practical price. Comprehensive retrieval logging at agentic volumes carries real storage and retention costs, so institutions should scope it deliberately and tier log detail and retention periods by the risk of the decision, rather than defaulting to keeping everything forever.

3. The Jury Pattern: Governing the Output Side

3.1 What it is

The Jury pattern applies a principle every risk officer already enforces on humans: the person who does the work does not approve the work. In an agentic system, this means the output of an implementing agent is evaluated by one or more independent critic agents, and an orchestrating component, the judge, presides over their findings and rules on what holds up.

The pattern's originators are explicit about the courtroom framing: many teams currently ask a single model to act as judge and jury at once. Separating the roles, with critics as the jury weighing evidence on its own merits and the orchestrator as the judge ruling on it, restores the segregation of duties that a single self-reviewing model collapses.

3.2 Diversity of opinion as a design principle

The most distinctive element of the pattern is the deliberate mixing of model families among the critics. Models from different vendors are trained on different data, shaped by different objectives, and tuned toward different behaviors, so they notice different risks and failure modes, much as a human review panel benefits from a mix of product, architecture, and operations perspectives. One documented implementation enforces this with a simple forcing function that keeps reviewers off the same model family, and industry best-practice

guidance echoes the underlying instinct: have a fresh model try to refute a result, so the agent that did the work is not the one grading it.

This is directly analogous to model risk management doctrine in banking. US supervisory guidance on model risk management (SR 11-7) makes "effective challenge" a central requirement, defining it as critical analysis by objective, informed parties who can identify a model's limitations and assumptions, and it expects those reviewers to be independent of the model's developers. The jury pattern is that principle rendered in software: a jury of critics from one model family is a review panel staffed entirely by the author's teammates. One caveat deserves stating: models from different vendors can still share training data and architectural lineage, so family diversity reduces correlated blind spots rather than eliminating them. It is a mitigation, not a guarantee of independence. The diversity requirement does carry procurement weight, though. Each additional model family means another vendor relationship to maintain, with its own contracting, operational infrastructure, and compliance documentation. For institutions with strict vendor management regimes this can be a significant implementation barrier, so a pragmatic reading of the pattern is two well-chosen families rather than many, with the vendor-management cost weighed explicitly against the independence benefit.

3.3 Adoption does not require the full courtroom

Practitioners advise starting small: a single critic on a different model family from the one that produced the plan, with explicit permission to disagree, already captures much of the value. The full jury, with multiple critics, family diversity enforced, and an orchestrator with authority to rule, is the destination rather than the entry ticket. The non-negotiable element at every stage is separation of responsibilities: the critic, the orchestrator, and the implementer must each be able to act without being overruled by the same model that proposed the work.

The pattern's costs are real and should be stated plainly: each independent review multiplies inference spend and adds latency, and disagreement among critics requires an adjudication policy (majority, severity-weighted, or judge's discretion). Practitioners are still debating when model disagreement adds signal versus noise. The pragmatic answer is to reserve the full jury for outputs whose failure cost justifies it, like client-facing communications, regulatory submissions, and production code changes, and use lighter review for low-stakes work. Review policy should also anticipate adversarial scenarios. If the critics suspect prompt injection, poisoned context, or deliberate manipulation of the pipeline, the correct behavior is not a majority vote but escalation: fail closed, quarantine the output, and route the case to human review with the full provenance record attached.

4. Two Patterns, One Control Chain

The patterns are frequently discussed separately, but their real power is compositional. Together they bracket the agentic pipeline with controls at both ends:

> *Librarian → curated, provenance-tracked inputs → Implementing agent(s) → draft output → Jury → adjudicated, independently reviewed result*

This composition matters because the two failure modes compound each other. A jury reviewing an output built on stale or polluted context can only catch what is visibly wrong; it cannot recover information the implementer never saw. Conversely, a well-fed implementer with no independent review will occasionally produce confident, well-sourced, wrong answers that ship unchallenged. Input curation without output review is trust without verification; output review without input curation is verification of work built on sand.

One design decision determines whether the composition actually delivers: the jury must see the librarian's provenance metadata, not just the finished output. Critics that receive sources, versions, retrieval timestamps, and staleness flags can verify that claims trace back to authoritative, current material. Critics that only receive the final draft can judge plausibility but not provenance, and much of the librarian's value gets lost at exactly the review stage where it matters most. Put simply, the control chain is only as strong as the metadata that flows down it.

Failure mode addressed	Context pollution: irrelevant, stale, or excessive information degrading agent reasoning	Self-grading bias: the model that produced the work also evaluating it
Where it sits	Upstream: controls what enters the context window	Downstream: controls what leaves the system as an accepted output
Core mechanism	Deliberate curation and routing: load only what the task requires; flag staleness; decide the right source	Independent multi-critic evaluation, ideally across model families, with an orchestrator ruling on findings
Governance analogue	Records management and data lineage controls	Segregation of duties; maker–checker controls
Primary cost	Routing logic build and maintenance; retrieval latency	Multiplied inference cost and latency per reviewed output

Table 1: The two patterns compared. Note how each maps to a control regulated institutions already operate for human processes.

For leaders framing an AI governance narrative internally, this table is the story: each control point in an agentic system has a direct analogue in a control the institution already runs, so adoption becomes a translation exercise rather than a leap into an unfamiliar control philosophy.

5. Implications for Regulated Financial Services

Three implications deserve particular attention from technology and risk leaders in banking, insurance, and wealth management.

5.1 Architecture is becoming the unit of model risk review

Model risk frameworks were written for discrete models with defined inputs and outputs. Agentic systems are pipelines of models, tools, and routing decisions. The Librarian and Jury patterns give validation teams identifiable control points to test: Was the retrieval policy followed? Were the critics independent? Was the adjudication logged? This is what named frameworks already presuppose: SR 11-7's demand for effective challenge and independent validation, the Govern–Map–Measure–Manage structure of the NIST AI Risk Management Framework (2023), and the human-oversight duties in Article 14 of the EU AI Act all assume you can identify where a system is controlled and show evidence that the control operated. An agentic system built on these patterns is reviewable against that expectation; a monolithic agent with implicit retrieval and self-review is not.

5.2 The audit trail comes for free, if you design for it

Both patterns naturally generate the artifacts an examination requires. The librarian logs what was retrieved, from where, and why. The jury logs each critic's findings and the judge's ruling. Institutions that adopt these patterns early will find that explainability and auditability are properties of the architecture rather than an after-the-fact reconstruction exercise. Three practical caveats apply here. Retrieval and review logs will routinely capture client data, so they need retention policies and privacy controls of their own. Access to the logs has to be governed as carefully as access to the underlying systems. And producing examination-ready artifacts from raw logs is an operational workload that should be budgeted for, not assumed.

5.3 Vendor concentration risk now has an architectural mitigation

The jury's insistence on cross-family critics has a second-order benefit: it forces the institution to maintain operational fluency with multiple model providers. That is a hedge against vendor concentration risk that most AI strategies currently address only on paper.

6. Adoption Roadmap

A pragmatic sequence for an institution starting from conventional RAG-plus-single-agent deployments:

The phases are deliberately incremental. Both patterns tolerate partial adoption; both begin paying back at the first step.

Each phase also needs a named owner. In most institutions the librarian is owned by the data or platform engineering team, with data governance setting source hierarchies and staleness policy. The critic and review process is usually operationalized by whichever function is accountable for output quality, often QA or model risk. The adjudication policy and the judge role belong jointly to technology and risk, since a ruling is both an engineering event and a control decision. Assigning these owners in Phase 0, before any tooling is built, keeps both patterns from becoming orphaned infrastructure.

7. Conclusion

The institutions that will operate agentic AI safely at scale are not the ones with the largest model budgets. They are the ones that recognize a familiar truth in a new costume: trustworthy systems are built from independent checks and disciplined information handling, not from trusting any single component, human or machine, to police itself.

A note on economics: the honest answer to "what is the return?" is that it is measurable but deployment-specific. The break-even logic is simple enough. Independent review pays for itself where the cost of a shipped failure, multiplied by how often failures occur, exceeds the added inference and latency spend, which is why the roadmap insists on measuring catch rates from Phase 2 onward. High-stakes output types in regulated institutions, such as client communications and regulatory submissions, tend to clear that bar quickly. Low-stakes internal drafting often does not, and should not carry a full jury. The patterns' incremental structure exists precisely so each institution can find its own break-even point with its own data, rather than leaning on an industry average.

The Librarian pattern brings records-management discipline to what an agent reads. The Jury pattern brings maker–checker discipline to what an agent ships. Together they convert

agentic AI from an unreviewable black box into a control chain your auditors, your validators, and your regulators can actually examine. That conversion, more than any model upgrade, is what will separate durable enterprise AI programs from expensive pilots.

About the Author

Smit Jani is a management consultant specializing in AI governance, responsible AI, and enterprise data strategy in regulated financial services, with engagements spanning GenAI adoption, audit remediation, and enterprise LLM platform delivery at major North American financial institutions. He holds the CAITL certification, writes the Human Centric AI newsletter, and is the author of the forthcoming book Human in the Lead.

References

Note: the two patterns themselves are drawn from publicly documented practitioner accounts whose vocabulary is still evolving, so readers should verify implementation details against those primary sources. The governance frameworks and research cited above (SR 11-7, the NIST AI Risk Management Framework, the EU AI Act, and the referenced studies) are established, independent references included to ground the analogies; they are not claimed to endorse these specific patterns.