WISSEN TECHNOLOGY

AGENTIC AI IN FINANCIAL SERVICES

The infrastructure that unlocks the value – and contains the risk

The non-obvious challenges of putting agents into production, and what good looks like.

Prepared By

Natarajan K Venkatasubramaniam

Executive Director, Wissen Technology

July 2026

DELIVERING 2X IMPACT

www.wissen.com

Table of Contents

Executive summary

Almost every financial institution is now building with artificial-intelligence (AI) agents; very few run them at scale in production. Recent 2026 surveys put the gap starkly: only about one in ten agent pilots ever graduate to production, and the blockers cited are evaluation, governance, data and security — not model quality[1]. Financial services is, if anything, ahead of most sectors in getting agents live, which makes the discipline of production the competitive question rather than an afterthought. That gap is the subject of this paper. It is also, we think, the realistic expectation. As Andrej Karpathy has argued, this is not the “year of agents” but the “decade of agents”: moving from a demo that works most of the time to a system that works reliably is slow, unglamorous engineering — his “march of nines,” in which each additional nine of reliability costs roughly as much as the one before it. In a regulated institution, where a wrong action is an incident rather than a retry, those further nines are the whole point.

The thesis is simple and, we find, under-appreciated: with agents, the model is no longer the hard part. The value and the risk have moved down a layer — into the action, control and governance plane. A copilot advises; an agent acts. The moment software can move money, amend a trade, refresh a KYC record or submit a regulatory report on its own initiative, the binding constraints become authorization, reproducibility of the decision path, deterministic control of consequential steps, agent identity, and the integration surface itself. The industry is racing to improve prompts and models. The value is unlocked one plane down, and that is where expertise now pays.

One point is specific to this sector, and it is the optimistic one. The controls that make financial services feel like hard ground for autonomy — model-risk management, information barriers, dual control, immutable audit, three lines of defense — are not obstacles to agentic AI; they are the scaffolding it requires. A firm that builds agents to satisfy them does not merely reach production, it ends up with a governed, reusable platform that is hard to copy. Financial services is not the hardest place to deploy agents. It is the proving ground, and the discipline is the moat — which is also why the blueprint extends, at the end of this paper, to every other regulated sector by swapping the overlay rather than rebuilding the engine.

This paper does not restate what the industry already knows. It catalogs the non-obvious challenges of agentic infrastructure at a financial institution — the ones that surface only when a pilot meets real entitlements, real controls and real scale — and for each one states plainly how we address it and what good looks like. It then gives a reference architecture, works one agent end to end — a reporting-break remediation agent assembled from portable Skills — and shows how to turn a backlog of agents into a compounding capability.

Who this paper is for

Putting agents into production in a regulated institution is not one leader's problem; it sits across six. Each will find the question they are accountable for answered here.

If you lead…	The question you are accountable for	Where the paper answers it
Technology / AI	How do we put agents into production safely, and at a cost that makes sense?	The challenge catalog, reference architecture and the worked example (Sections 2–4).
Operations	Can agents clear our exception, break and case queues reliably — at the deadline — without creating new operational risk?	The map of operational workflows (§ 1.1), the reporting-break agent worked end to end (Section 4), and orchestration and observability for unattended runs (§ 2.7, § 2.9).
Compliance / Risk	Can we evidence, reproduce and control what an autonomous system does — to a model-risk and audit standard?	Model risk and reproducibility (§ 1.2, § 2.1–2.3), evaluation (§ 2.8) and the audit trail (Section 4).
Finance / Accounting	What does running agents actually cost, what is the payback, and does automated activity stay inside financial controls?	The economics and cost of error (§ 1.1), cost per completed task and budgets (§ 2.10), and the auditable trail behind financially material actions (§ 2.3, Section 4).
Business	Where is the value, how quickly can we capture it, and what does getting it wrong cost?	The value map and economics (§ 1.1), trajectory cost (§ 2.10) and the first-90-days path (Section 6).
Legal	When an agent acts, who is accountable, on whose authority, and how fast can it be stopped?	Delegated authority and agent identity (§ 2.4–2.5), dual control and immutable audit (§ 2.3, Section 4).

1. The financial-services opportunity

Before the challenges, the prize. Agentic AI is not a single use case in financial services; it is a pattern that recurs at dozens of points across the value chain — wherever high-volume, judgment-heavy, multi-system work runs against a deadline and a control. This section maps where the value is, argues that this sector is the proving ground rather than the hard case, and then explains why agents change the engineering problem.

1.1 Where agents create value across the value chain

The shape is consistent across the sector: a queue of exceptions or cases, each requiring a person to gather context from several systems, apply judgment against a policy, and take an action that must be evidenced. That is precisely the shape an agent fits — and precisely where the control burden is highest. Most of these seams sit in operations — post-trade processing, reconciliation, exception and case management — which is where the earliest and most defensible agentic value in financial services is concentrating. The map below is not exhaustive; it marks the richest seams.

Business area	Representative agentic use cases	Value driver	The binding control
Capital markets — post-trade	Trade-break and reconciliation remediation; regulatory-reporting break remediation (the Markets in Financial Instruments Regulation, MiFIR; the European Market Infrastructure Regulation, EMIR; and the Securities Financing Transactions Regulation, SFTR); corporate-actions processing; margin and collateral dispute resolution.	T+1 (trade date plus one) timeliness; fewer fails; analyst leverage.	Audit; resubmission authority; dual control.
Capital markets — front / pre-trade	Pre-trade compliance and eligibility checks; research summarization; request-for-quote (RFQ) and quote drafting.	Speed; coverage; consistency.	Best execution; market abuse (the Market Abuse Regulation, MAR); information barriers.
Buy-side / asset management	NAV (net asset value) oversight and IBOR (investment book of record) reconciliation; mandate and guideline-breach triage; trade-allocation exceptions.	Oversight coverage; error reduction.	Valuation governance; fiduciary duty.
Retail & commercial banking	KYC (Know Your Customer) refresh and periodic review; AML (anti-money-laundering) alert triage and case build; sanctions-hit adjudication; complaints handling; dispute and chargeback resolution.	Analyst leverage against vast alert volumes; cycle time.	SAR (Suspicious Activity Report) obligations; Consumer Duty; adverse-action notice.
Wealth & advisory	Suitability and appropriateness checks; portfolio-review preparation; client onboarding.	Adviser leverage; consistency.	Suitability; fiduciary duty; record-keeping.
Insurance	Claims triage and adjudication; underwriting assistance; subrogation identification; fraud investigation.	Cycle time; leakage reduction.	Fair-claims handling; conduct rules.
Payments	Exception and return handling; fraud-case investigation; reconciliation.	Straight-through rates; loss reduction.	Scheme rules; fraud-liability rules.

The economics are not marginal. In anti-money-laundering alone, rule-based transaction monitoring produces false-positive rates above 95% — by some measures near 98% — so the overwhelming majority of the analyst effort a firm funds is spent clearing noise, and roughly two-thirds of financial-crime-compliance cost sits in KYC and customer due diligence[2]. The cost of getting it wrong is larger still: global penalties for AML, KYC and sanctions failings totaled about USD 3.8 billion in 2025, the single largest of them about USD 985 million[3]. Every seam in the table has that profile — large manual volumes, a deadline, and a control that makes errors expensive. That combination is exactly what disciplined agentic automation is built to address.

Figure 1 — A financial-services agent value map: where to start (illustrative)

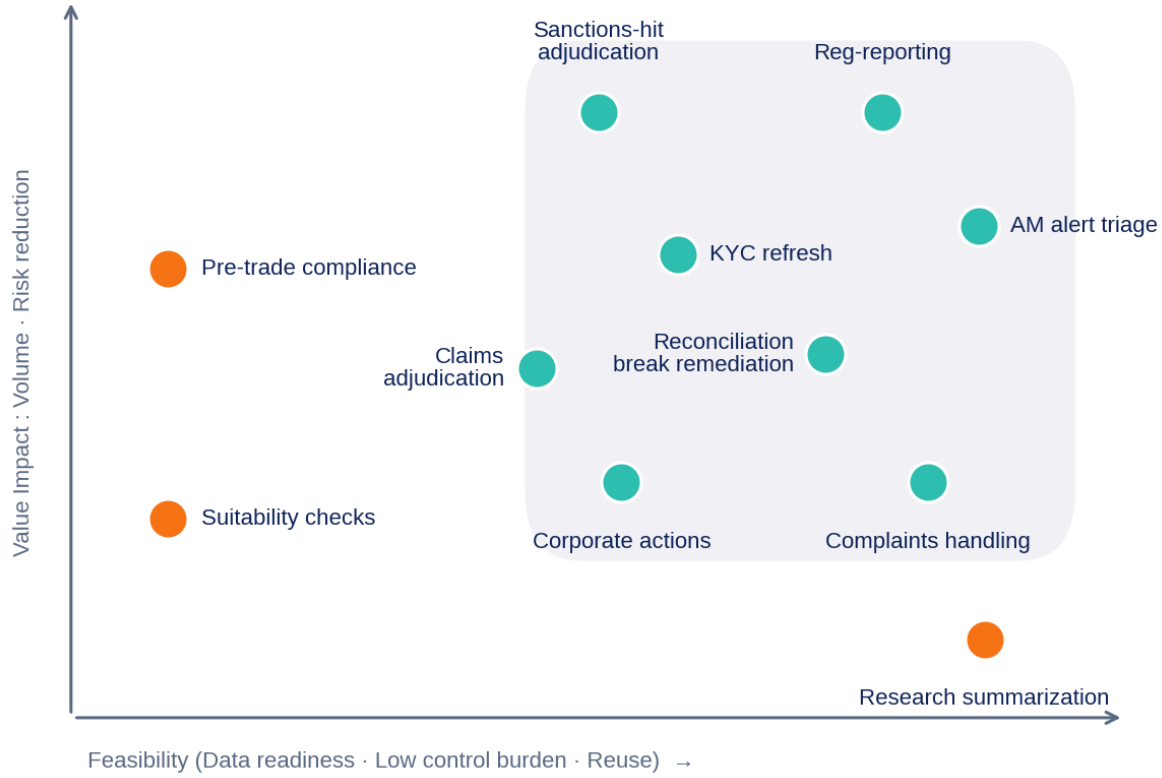


Figure 1 — a way to prioritize: agentic use cases plotted on value against feasibility; the routine, high-volume, high-control seams (top-right) are where to start.

1.2 Why financial services is the proving ground, not the hard case

The reflex is to treat financial services as the hardest place to deploy autonomy — too regulated, too much at stake. We take the opposite view, and it is the more useful one. The controls that make the sector feel like hard ground — model-risk management (SR 11-7; the Prudential Regulation Authority's SS1/23)[4], information barriers, dual control, immutable audit, the three lines of defense — are not obstacles to agentic AI. They are the scaffolding it requires anyway. An agent that must justify every action to a validator, act within a human's entitlements and leave a replayable trail is simply an agent built correctly. A firm that builds to these constraints does not merely reach production; it ends up with a governed, reusable agent platform a less-disciplined competitor cannot easily copy. **In this sector the discipline is the moat.**

Concretely, each defining feature of the financial-services control environment maps onto a specific piece of the architecture in Section 3 — which is why building for the sector and building well are the same exercise.

Financial-services force	What it demands of an agent	Where the architecture answers it
Model-risk management (SR 11-7 / SS1/23)	Reproduce and validate every decision — including the path taken.	Trajectory store; the agent-as-system in the model inventory (§ 2.3).
Information barriers / Chinese walls	Knowledge and memory must not cross entitlement boundaries.	Barrier-scoped memory; filtered, entitlement-aware retrieval (§ 2.6, § 2.12).
Three lines of defense	Clear ownership and independent challenge of the agent.	Agent identity and policy engine; independent evaluation and observability (control & assurance planes).
Dual control / four-eyes	No unilateral high-consequence action.	Authorization gate with step-up approval (§ 2.4).
Immutable audit & record-keeping	Attributable, tamper-evident evidence of what was done and why.	Decision trace and evidence generation (§ 2.3; Section 4).
Deadline compression (T+1 / T+0)	Bounded, reliable completion inside the window.	Bounded horizons and the deterministic spine (§ 2.1, § 2.2).

1.3 Why agents change the problem

An agent is not a bigger model; it is a **loop** — plan, call a tool, observe, decide, repeat — with the authority to change the state of the world. The canonical decomposition, set out in Lilian Weng's account of agents powered by large language models (LLMs), is planning plus memory plus tool use; each of those becomes a control problem the moment the agent can act. Two consequences follow that most benchmarks miss.

First, reliability becomes multiplicative. A single call at 95% reliability is fine. Chain twenty dependent steps and, if the errors were independent, you would be at roughly 36% — not because the model got worse, but because the horizon got longer. Real errors are rarely fully independent: correlation can soften the maths or, when one bad step poisons the next, make it worse. The precise number is illustrative; the direction is not — reliability falls as the horizon grows. Teams pilot a five-step agent that works beautifully, scale it to a twenty-five-step reconciliation, and watch reliability quietly collapse.

Figure 2 — reliability is multiplicative: the compounding error cliff

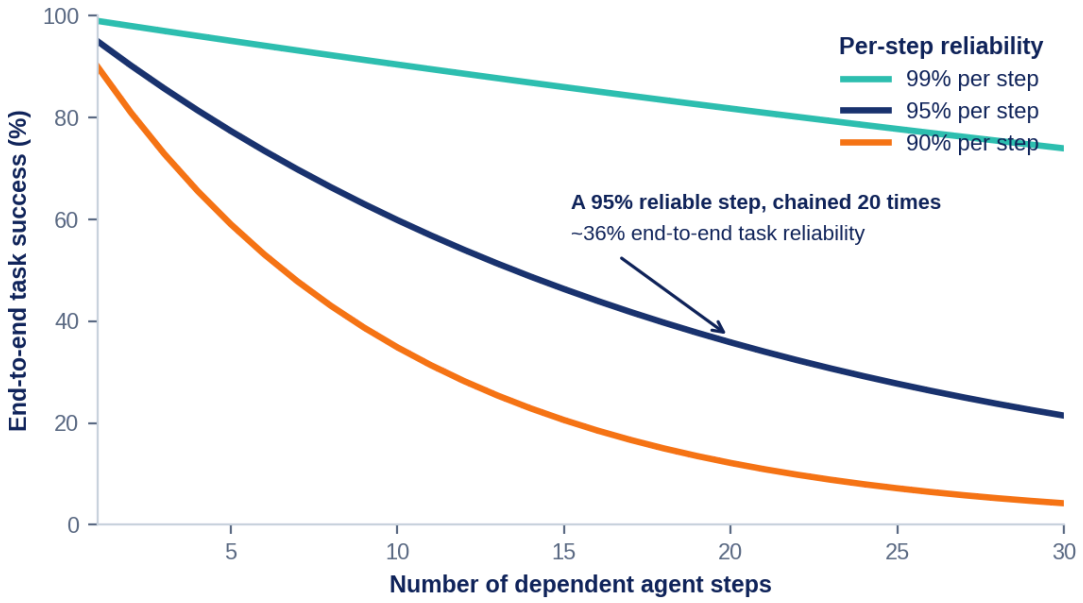


Figure 2 — chaining steps multiplies error; a 95%-reliable step run twenty times yields only ~36% task reliability.

Second, the failure surface moves from words to actions. A copilot that hallucinates produces a wrong sentence a human can catch. An agent that hallucinates can execute a wrong action before any human sees it. Everything in the catalog that follows exists to make autonomous action safe, reproducible and economical in an environment where a single wrong action is a regulatory event, not a typo.

2. The challenge catalog

The distance between a compelling pilot and a production system is not closed by a better model. It is closed by a specific set of challenges that appear only when an agent meets real entitlements, real controls and real scale. This section is that set. We begin, for completeness, with the challenges the industry has already converged on — the table stakes — then spend the rest of the paper on what remains genuinely hard.

A word on how to read the table. Every item below is necessary and none is where advantage is won. The point of view is in the third column: each “solved” problem leaves a residue the standard remedy does not touch, and in a financial institution that residue is where the incidents happen. It is also the bridge into the non-obvious catalog that follows — each residue is taken up in a later section, cross-referenced.

Well-known challenge	Standard remedy (table stakes)	The financial-services residue the remedy leaves
Hallucination / weak grounding	Retrieval-augmented generation (RAG) with mandatory citations; confidence thresholds that abstain; human review for high-stakes outputs.	The dangerous failure is not the obvious fabrication but the plausible one — a citation to a real but inapplicable rule. Groundedness must be scored on applicability, not the presence of a source (§ 2.8).
Non-determinism (per call)	Temperature zero where stability matters; structured / constrained outputs; pinned model versions.	Determinism at the call does not survive the loop: the same task still takes different paths run to run. Reproducibility must be engineered at the trajectory level (§ 2.3).
Data privacy — PII / MNPI leakage	Classify data before it reaches the model; barrier and tenant isolation; data-loss-prevention (DLP) on prompts and outputs; no-train terms.	Classification guards data at rest and in prompts, but agents open new leakage paths — shared memory and inter-agent hand-offs — that bypass DLP entirely (§ 2.6).
Prompt injection (baseline)	Treat external and retrieved content as untrusted; input sanitization; least-privilege tool access.	Filtering assumes a single turn. In an agent, injected content persists in memory and re-surfaces later, and the real exposure is the action it triggers, not the text (§ 2.4).
Cost and latency	Difficulty-based model routing; caching; right-sizing; batch where the workload is not real-time.	Per-call control misses the dominant term: agent cost is driven by trajectory length and re-planning, and one task can vary tenfold run to run (§ 2.10).
Explainability and audit	Decision logging; entry in the model inventory; alignment to SR 11-7 model-risk governance.	Logging input and output satisfies a model review; it does not let a validator reconstruct why the agent chose its path — what an agent audit actually demands (§ 2.3).
Bias and fairness	Fairness testing against protected classes as a standing check, not a one-time audit.	Testing outputs misses process bias: an agent can reach a fair decision by a path that systematically disadvantages a segment — e.g. which cases it escalates. Grade the trajectory (§ 2.8).
Data quality and readiness	Data governance, lineage and curation before deployment — the most common blocker to production.	Document readiness is not the bar. Agents need governed, typed access to live system state — positions, limits, entitlements — which most data programs have not built (§ 2.11).
Model / vendor drift	Version pinning plus a regression suite that gates every model, prompt or index change.	Pinning the model is not enough once tools, prompts and skills also drift. The unit under version control is the whole agent-as-system, not the model (§ 2.3).

From table stakes to the real work. The residue above is not a set of loose ends; it is a map. The challenges that separate production from pilot fall into four areas, each a different way an agent breaks what worked in the demo:

- **Reliability and control** — an agent is a chain of dependent steps with authority to act, so small per-step error compounds and probabilistic control flow becomes a liability.
- **Authorization, identity and data** — once software acts on its own initiative, the question is no longer what it can read but what it may do, on whose authority, and under what identity.
- **Composition and assurance** — when agents call agents and run unattended, failure becomes emergent and silent, and correctness must be judged on process, not just outcome.
- **Economics and the real bottleneck** — at scale the binding constraints are trajectory cost, the integration surface and retrieval quality, not the model.

Figure 3 — the challenge catalog at a glance: four themes, twelve non-obvious challenges

Figure 3 — the challenge catalog at a glance: four themes, twelve non-obvious challenges.

For each challenge we state the problem as we meet it in the field, how we address it, and what “good” looks like once it is solved. We begin with reliability and control.

2.1 Reliability and control

THE COMPOUNDING-ERROR CLIFF

The non-obvious problem. Agent reliability is not a property of the model; it is, to a first approximation, the product of every step's reliability (exactly so only when errors are independent — a simplification, but the direction holds regardless). The consequence is counter-intuitive: making the model 2% better barely moves a long-horizon agent, but **removing three steps** from the trajectory moves it dramatically. Firms tune the wrong variable — model quality — when the lever is horizon length.

How we address it. Decompose long horizons into short, independently verifiable sub-tasks; insert **deterministic post-condition** checks between steps (assertions against systems of record, not the model checking itself); make each step idempotent and resumable; prefer plan-then-verify over open-ended reasoning loops for consequential work; bound trajectory length and fail closed when it is exceeded.

What good looks like. Every workflow has a bounded, measured horizon; each consequential step has a machine-checked post-condition; a failed check halts and escalates rather than proceeding on corrupted state; and reliability is reported **end-to-end per completed task**, never as a per-call number.

THE DETERMINISTIC SPINE — DON'T LET THE MODEL OWN CONTROL FLOW

The non-obvious problem. The instinct is to let the agent reason freely and act at will. For consequential financial-services actions this is exactly backwards. Language models are excellent at judgment (which break category is this? does this narrative satisfy the policy?) and unreliable at control flow and state management. Handing the model the control flow is handing a probabilistic process a deterministic job.

How we address it. Invert the architecture. A deterministic workflow engine — a state machine — owns control flow and every consequential transition; the model is invoked only at defined decision nodes to make bounded judgments the deterministic layer then validates and executes. In the idiom that has become common in agent engineering, the agent proposes; deterministic code disposes. This is the disciplined form of Anthropic's guidance in Building Effective Agents — prefer the simplest composition that works, and reach for open-ended autonomy only when the task genuinely demands it. Posting a correction, releasing a payment or submitting a filing is a typed, tested code path with explicit pre- and post-conditions, never a string emitted by the model.

What good looks like. Consequential actions never originate directly from model output; the workflow is unit-testable independent of the model; the model's remit is scoped to classification, extraction and judgment at named nodes; and control flow is inspectable and version-controlled.

REPRODUCIBILITY OF TRAJECTORIES, NOT OUTPUT

The non-obvious problem. Model-risk governance (SR 11-7) and audit require you to reproduce a decision. For a single model you version input and output. An agent takes a different path on every run — different tool calls, retries, intermediate states. Reproducing the answer is not enough; a validator needs the path. Teams that log only input and output are blindsided the first time the second line asks why the agent did what it did.

How we address it. Treat the **trajectory as a first-class, versioned artifact:** the plan, every tool call and result, intermediate state, the model and prompt version at each step, the branch points, and the entitlements context at the moment of each action. Store it replayable and diffable; pin models per step.

What good looks like. Any agent decision is replayable step-by-step months later; two trajectories can be diffed; the trajectory store is the audit system of record; and the model inventory holds the **agent-as-system** — its tools, guardrails and action space — not merely a model card.

2.2 Authorization, identity and data

THE CONFUSED DEPUTY — AUTHORIZE ACTIONS, NOT JUST DATA

The non-obvious problem. The dangerous shift from copilot to agent is that the agent acts. The naive pattern gives it a service account with broad entitlements — so a single prompt injection or reasoning slip can move money, amend a trade or file a report. The subtlety financial institutions miss: access control over data is not the same as authorization of actions, and an agent must never be able to exceed the authority of the human it acts for.

How we address it. Propagate the user's identity and entitlements to the agent through on-behalf-of / delegated-authority tokens, so effective authority is the intersection of the agent's and the human's. Authorize every tool invocation — not just reads — through a central policy engine. Issue narrow, task-scoped capability tokens. Note a 2026 finding that matters in practice: time-to-live revocation **fails at agent execution speed**[5]; tokens must be attenuated and revocable by invocation count, not merely by clock. Require dual control for high-consequence actions, and treat every retrieved or external input as an injection vector at the action boundary. Simon Willison's "lethal trifecta" names the danger precisely — an agent that combines access to private data, exposure to untrusted content, and the ability to communicate externally can be turned into an exfiltration tool by a single poisoned input; the defense is to ensure at least one leg is always missing on any consequential path, which in practice means an egress allow-list at the action gateway.

What good looks like. An agent cannot perform an action the requesting human could not; every action is centrally authorized and attributable to (agent, on-behalf-of human, task); capability tokens are least-privilege and invocation-bound; high-consequence actions demand step-up approval.

Figure 4 — Solving the confused deputy: delegated authority + per-action authorization

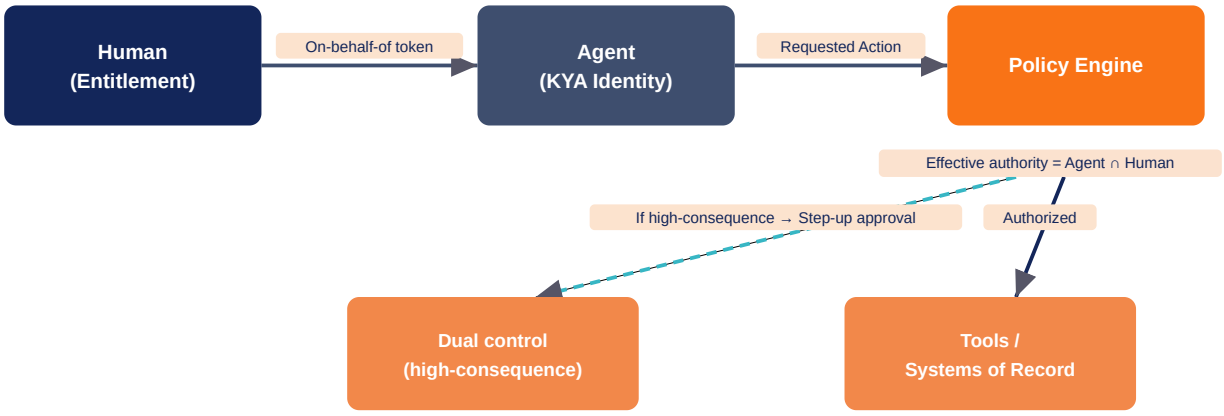


Figure 4 — delegated authority plus per-action authorization: the agent is bounded by the human's own entitlements.

AGENTS NEED FIRST-CLASS IDENTITY ("KNOW-YOUR-AGENT" KYA)[6]

The non-obvious problem. Institutional identity and access management (IAM) is built for humans and static service accounts. Agents are autonomous, non-human actors that reason and act — and running them on shared service accounts destroys attribution and makes revocation impossible. Regulators are already moving from **Know-Your-Customer to Know-Your-Agent:** verifiable agent identities linked to a legal entity. Firms without an agent registry will not be able to answer "which agent, acting for whom, did this — and on whose authority?"

How we address it. Give every agent (ideally every agent instance or task) a first-class identity in IAM, with its own entitlements and full lifecycle — provision, rotate, revoke. Maintain a registry of agents in operation with their capabilities and grants. Stamp every action with agent identity, the on-behalf-of principal and the task ID.

What good looks like. Agents are governed identities with least-privilege grants and instant revocability; a live registry answers who and what is running; every action is attributable end-to-end and defensible to a regulator.

MEMORY IS AN ATTACK SURFACE AND A BARRIER-GOVERNANCE PROBLEM

The non-obvious problem. Agents need memory to be useful across a workflow — but persistent, shared memory introduces two hazards the industry rarely names. First, memory poisoning: a malicious or simply erroneous entry persists and silently corrupts future decisions — a durable prompt injection. Second, barrier leakage: shared memory can carry MNPI or client data across an information barrier or entitlement boundary. Shared “long-term memory” across agents is, at a financial institution, a compliance landmine.

How we address it. Scope memory by information barrier and data classification; attach provenance and a time-to-live to every item; validate and quarantine writes so unverified external content never enters durable memory; never share memory across barriers or entitlement domains; and subject memory reads to the same entitlement checks as live data.

What good looks like. Memory is classified, provenance-tracked, expiring and barrier-scoped; a poisoned or stale entry cannot silently steer a decision; and MNPI cannot traverse a wall through memory.

2.3 Composition and assurance

MULTI-AGENT ORCHESTRATION — EMERGENT FAILURE AND HALLUCINATION PROPAGATION

The non-obvious problem. Compose agents and you inherit failure modes single agents don’t have: unbounded loops, agents “negotiating” indefinitely, runaway cost, and — most insidious — **hallucination propagation,** where one agent’s fabrication becomes another’s ground truth, laundering a guess into an authoritative input. Free-text hand-offs between agents are the vector.

How we address it. Use a supervisor/orchestrator with hard budgets (steps, tokens, wall-clock) and circuit breakers per agent. Make hand-offs **typed and schema-validated** — structured artifacts, not prose — so every claim is checkable at the boundary and carries provenance. Ground every inter-agent claim in a verifiable system-of-record read, never in another agent’s assertion.

What good looks like. Every agent runs under a budget and a circuit breaker; hand-offs are typed and validated; no agent treats another’s unverified output as fact; and the orchestrator can halt a runaway trajectory before it spends or acts.

EVALUATING AGENTS MEANS GRADING THE PROCESS, NOT JUST THE ANSWER

The non-obvious problem. For a single model you score the output. For an agent, a **correct answer reached by a non-compliant path is a failure** — the agent that produced the right remediation but touched data it wasn't entitled to, or skipped a required control, has failed even though the answer is right. And you cannot safely test agents in production, because they act.

How we address it. Evaluate the trajectory: a large-language-model judge ("LLM-as-judge") over the trace plus deterministic policy-adherence checks; reward process, not just outcome. As practitioners such as Chip Huyen and Shreya Shankar have stressed, evaluation is the discipline that separates a demo from a product — and the judge itself must be validated ("who validates the validators?") against human-labeled ground truth before it is trusted. Build a simulation sandbox with mocked tools and systems of record to exercise agents — including adversarial and injection trajectories — before and continuously after release, an evaluator–optimizer loop in Anthropic's terms. Regression-test against recorded trajectories on every change.

What good looks like. Agents are validated on process and outcome; a realistic sandbox exists; adversarial trajectories are part of the suite; a policy violation fails the eval even when the answer is correct; and regression runs on every prompt, model or tool change.

OBSERVABILITY — FAIL SILENTLY

The non-obvious problem. A microservice that fails throws an error. An agent that fails often returns a confident, wrong, expensive answer. Standard application performance monitoring (APM) — latency and error rate — is blind to this. The failure is **semantic:** the agent is still "working," just no longer behaving.

How we address it. Instrument reasoning-level observability: distributed-tracing-style spans for each step (plan, tool call, result), semantic drift detection on the **distribution** of trajectories (step counts, tool-use patterns, escalation and abstention rates), and canary trajectories with known-good outcomes running continuously, plus cost and step telemetry per task.

What good looks like. Every step is traced and inspectable; behavioral drift is caught before it becomes an incident; canaries alarm on silent degradation; and you can always answer "what is this agent doing right now, and is it normal?"

2.4 Economics and the real bottleneck

TRAJECTORY LENGTH, NOT TOKEN PRICE, DOMINATES COST

The non-obvious problem. Procurement negotiates per-token price and misses that agent cost is driven by **trajectory length and re-planning.** A “cheap” agent that loops or re-plans can cost fifty to a hundred times a single call, and the same task can vary tenfold run-to-run. Optimizing the model’s unit price while ignoring trajectory efficiency is optimizing the wrong variable.

How we address it. Measure cost per completed task under a hard trajectory budget; route per step — small models for routine extraction and formatting, a frontier model only for planning and hard judgment; cache tool results and sub-plans; prune context aggressively, since agents accumulate context bloat across steps; and shorten horizons (which also raises reliability — see 2.1).

What good looks like. Cost is tracked per completed task with enforced budgets; most steps run on small models; trajectory length is monitored and bounded; run-to-run variance is contained; and the reported metric is **value per completed task,** not price per token — a figure Finance can budget, forecast and hold teams to.

THE REAL BOTTLENECK IS THE ACTION SURFACE, NOT THE MODEL

The non-obvious problem. The industry frames agent value as model capability. At a financial institution the binding constraint is the integration and entitlement surface — legacy systems without clean APIs, batch data, fragmented entitlements. An agent is only as capable as the sanctioned, typed actions it can take against systems of record. Retrieval over documents is not enough; agents need governed, structured access to **live state** — positions, limits, KYC status, case data.

How we address it. Build a governed action catalog: typed schemas over systems of record, each action entitlement-aware and audited, each independently testable. Treat the catalog as the product. Map it in the organizational value graph (Section 5) so agents and teams discover and reuse sanctioned capabilities instead of re-wiring integrations.

What good looks like. A curated, typed, entitlement-aware action catalog is the platform; new agents compose existing sanctioned actions rather than each team building its own integrations and controls; and the value graph shows where value concentrates and what already exists to reuse.

RETRIEVAL AT SCALE: WHEN THE VECTOR STORE BECOMES THE BOTTLENECK

The non-obvious problem. A retrieval-augmented pilot on a small corpus grounds beautifully; at production scale it quietly stops working, and the usual reflexes — more shards, bigger machines, a higher recall setting — often make it worse. Two facts financial institutions rarely plan for. First, approximate-nearest-neighbor (ANN) **recall falls as the corpus grows** — even at modest sizes — because dense high-dimensional space becomes noisy, and no amount of index tuning fully fixes it. Second, a financial institution's retrieval is almost always **filtered** — scoped by entitlement, information barrier, instrument, book or date — and filtered search degrades recall further as the filter tightens, whatever the engine. For an agent this compounds: every retrieval step inherits the degraded recall, and weak grounding at step three corrupts steps four through ten.

How we address it. Budget recall explicitly and hold it to an agreed floor; measure end-to-end retrieval latency, not just the vector-search call. Choose the index to the workload's filter selectivity and expose an exact-search fallback for critical, low-selectivity queries. Control memory and latency with quantization and disk-resident indexes at billion scale, and separate index-build from serving. Above all, make retrieval hybrid and graph-anchored: use the capability graph and metadata / entitlement filters to pre-select a candidate set, then vector-search within it — the reliable way to hold recall as the corpus grows[7].

What good looks like. Recall is a monitored service-level objective (SLO) with an agreed floor, held as the corpus grows; retrieval latency is measured end-to-end and bounded; entitlement-scoped queries do not silently lose recall; and the retrieval layer is hybrid — graph- and metadata-anchored — so scale sharpens precision instead of eroding it.

3. The agentic reference architecture

The catalog resolves into one architecture. It is deliberately opinionated about a single thing: a **deterministic workflow** spine owns control flow, and the model is invoked around it for judgment. Two planes wrap every action — a control plane (identity, authorization, governance) and an assurance plane (evaluation, observability, the trajectory store). Nothing consequential happens outside both.

Figure 5 — Agentic reference architecture; the deterministic spine owns control flow; control and assurance planes wrap every action

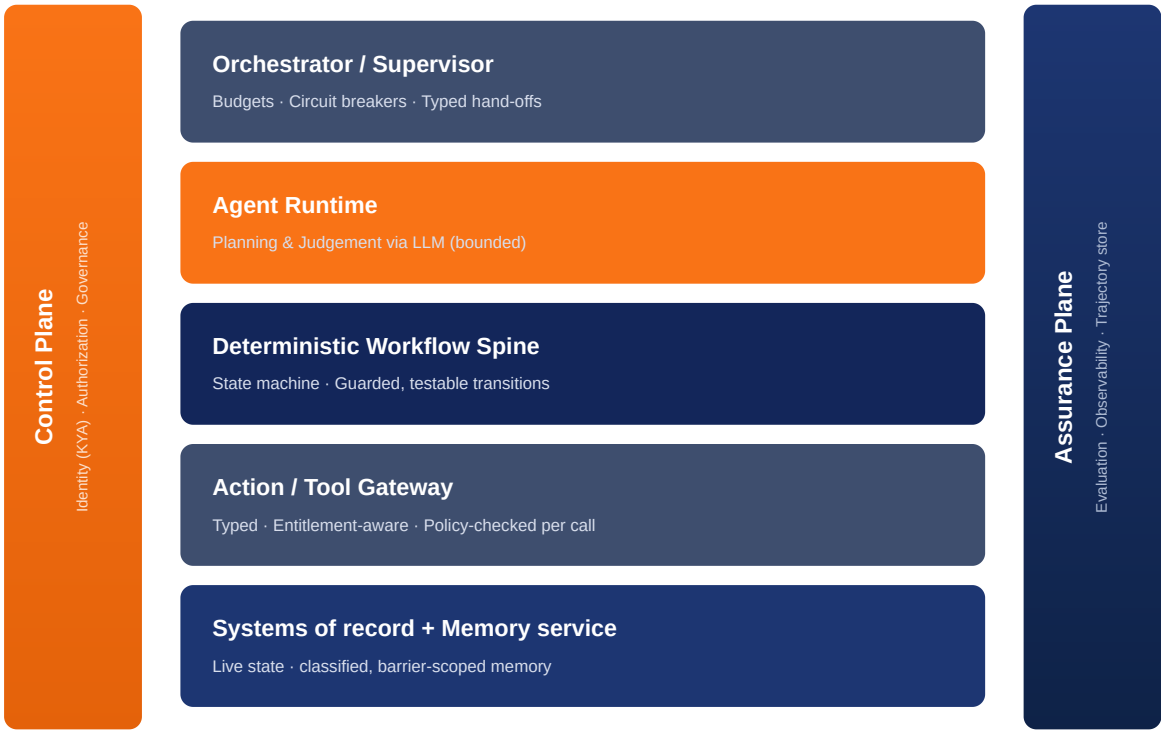


Figure 5 — the deterministic spine owns control flow; control and assurance planes wrap every action.

Component	Role	The requirement institutions under-build
Orchestrator / supervisor	Coordinates agents; enforces budgets and circuit breakers.	Typed, validated hand-offs so hallucination cannot propagate.
Agent runtime	Planning and judgment via the model, bounded.	The model proposes; it never executes consequential actions directly.
Deterministic workflow spine	State machine; consequential transitions as tested code.	Control flow is code, not model output — unit-testable without the model.
Action / tool gateway	Every tool call, typed and policy-checked.	Per-action authorization with delegated authority, not a broad service account.
Systems of record + memory	Live state and working memory.	Memory classified, provenance-tracked, barrier-scoped, expiring.
Control plane	Agent identity (KYA), authorization, governance.	First-class agent identity and a live agent registry.
Assurance plane	Evaluation, observability, trajectory store.	Trajectory-level evals and replayable traces for SR 11-7.

4. Worked example: the reporting-break remediation agent

Regulatory transaction reporting — MiFIR, EMIR and SFTR in Europe, their Commodity Futures Trading Commission (CFTC) and Securities and Exchange Commission (SEC) equivalents in the US — continually produces breaks: rejections where a submitted report fails validation, and reconciliation exceptions where the firm's record and the trade repository's do not agree. Remediation is high-volume, deadline-bound (typically T+1), judgment-heavy and touches many systems. That makes it an ideal agentic workload — and a demanding one, because a resubmission is an action taken with the regulator, not a draft. It is where every challenge in Section 2 becomes concrete.

4.1 Decompose the work into skilled sub-agents

The orchestrator owns the break lifecycle on the deterministic spine — detect, triage, root-cause, plan, correct and resubmit, evidence — and invokes a specialized Skill at each judgment node. We use "Skill" in the precise, open-standard sense: a versioned folder containing a SKILL.md (its trigger description and instructions) plus the scripts and reference resources a step needs. Skills load by **progressive disclosure** — only a skill's name and description sit in context until a task matches it, then its instructions load, then its reference files load only when a step reaches them.

Two things follow that a financial institution should care about.

- **It is model-agnostic by construction.** Agent Skills is an open standard adopted across the major model providers and agent runtimes[8], so the firm authors the "reporting-break-triage" skill once and runs it on whichever model it certifies — no rewrite to change model, and no lock-in.
- **It directly attacks the compounding-error and cost problems.** Progressive disclosure keeps each sub-agent's context lean and focused, which raises per-step reliability (Section 2.1) and lowers token cost (Section 2.10) at the same time.

Sub-agents run with **isolated context** under the orchestrator (the orchestrator-workers pattern), so one sub-agent's context bloat or error cannot contaminate another, and hand-offs between them are typed and schema-validated rather than free text — which is what stops one skill's guess from becoming the next skill's fact (Section 2.7). Tools are exposed through the Model Context Protocol (MCP), the open tool-connectivity standard[9]: read tools are broadly available, while the write tools — stage-correction and submit-resubmission — sit behind the action gateway with per-action authorization.

Skill (portable SKILL.md)	What it does	Bundled resources
reporting-break-triage	Classifies the break against the firm's break taxonomy.	Break taxonomy · field dictionary · validation rules
root-cause-analysis	Traces the break to source (a lapsed Legal Entity Identifier (LEI), reference-data mismatch, mapping error, late/duplicate, eligibility mis-scope).	Reconciliation-diff script · reference-data map
remediation-planner	Chooses the remediation: amend, cancel-and-correct, backfill reference data, re-flag eligibility.	Remediation playbook · rule-to-action map
reporting-evidence	Assembles the audit narrative citing the exact rule and field.	Evidence template · citation index

4.2 A skill, concretely

A skill is plain, portable and auditable — a domain expert can read it. The trigger description is what the agent reasons over to decide when to load it; the allowed-tools line is a control surface, constraining what the skill may invoke.

```
---
name: reporting-break-triage
description: >-
  Classify a transaction-reporting break against the firm break
  taxonomy. Use when a MiFIR/EMIR/SFTR report is rejected or fails
  reconciliation against the trade repository.
allowed-tools: [get-break, get-trade, get-reference-data]  # read-only
---

# Reporting-break triage
1. Load the break detail and the matched trade.
2. Compare reported vs expected fields (see references/FIELD-MAP.md).
3. Assign one taxonomy code; if confidence < 0.8 → abstain, escalate.
4. Return the typed triage contract. Do NOT propose a fix here.
```

4.3 The trajectory, end to end

Take a common case: a counterparty's LEI has lapsed and reference data was not refreshed, so the report is rejected. The agent's path is bounded and every consequential transition is deterministic.

- **Triage.** The triage skill classifies a counterparty-reference break; a deterministic post-condition confirms the code is a known taxonomy value, else it escalates.
- **Root cause.** The root-cause skill traces it to a lapsed LEI with stale reference data; a deterministic check validates LEI status against the authoritative source before proceeding.
- **Plan.** The planner proposes refresh-LEI plus cancel-and-correct of the affected report. The deterministic spine regenerates the report and runs it through the reporting engine's schema validation — a hard post-condition: nothing is submitted on a report that fails validation.
- **Authorize.** Resubmission is high-consequence, so it passes the authorization gate: the agent acts within the reporting officer's entitlements (delegated authority), a human approves under dual control, and the write executes on an invocation-bound capability token — not a standing service account.
- **Submit and evidence.** The action gateway submits to the trade repository; the evidence skill assembles the audit narrative citing the exact Regulatory Technical Standard (RTS) field and remediation code; the full trajectory is written to the trajectory store, replayable for the second line.

Figure 6 — Reporting-break remediation: skilled sub-agents for judgment, a deterministic spine for control

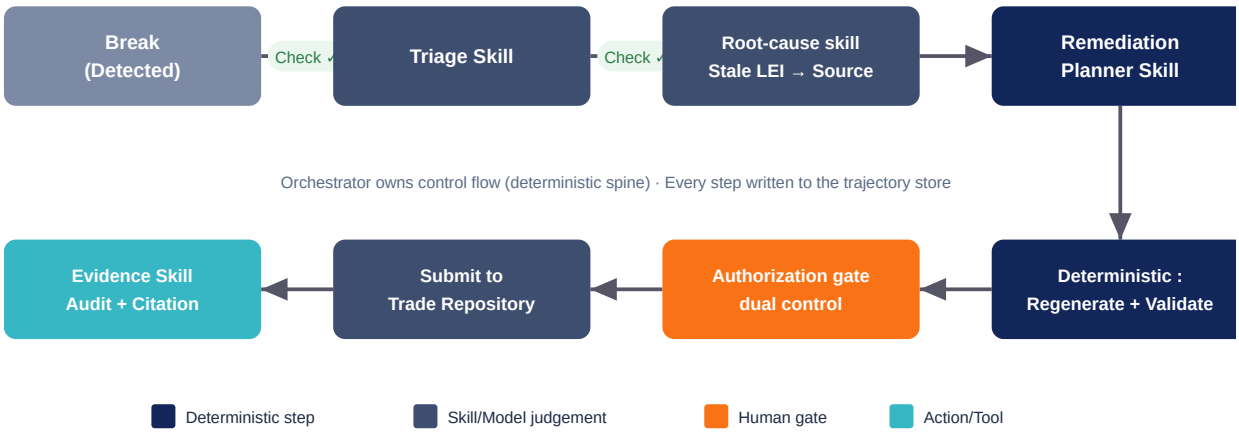


Figure 6 — skilled sub-agents supply judgment; the deterministic spine owns control flow and every consequential transition.

4.4 The artifacts that make it governable

The typed hand-off contract between sub-agents carries provenance, so no downstream skill treats an upstream guess as fact:

```
triage → root_cause hand-off {
  break_id, taxonomy_code: "CPTY-REF-LEI", confidence: 0.94,
  evidence_refs: [ tr_store://break/8842, sor://trade/55193 ],
  entitlements_ctx: { on_behalf_of: user_id, scope: [reporting.read] }
}
```

The trajectory-trace record is the audit system of record — the whole path, not just the answer:

```
trajectory {
  task_id, agent_id, on_behalf_of, break_id,
  steps[]: { skill, model_id+version, tool_calls[], result,
             post_condition: pass|fail, tokens, cost_usd },
  authorization: { action:"submit-resubmission", policy:"dual-control",
                   approver, token: invocation-bound },
  outcome, rule_citations[], replayable: true
}
```

4.5 Evaluation and economics

The agent is graded on process, **not just outcome.** A golden case fixes the expected classification, the expected remediation type, the entitlements it must not exceed, and the rule it must cite — and, decisively, that it must not submit without the dual-control gate.

```
golden_case {
  input: break(kind="LEI lapsed", instrument, cpty),
  expect: { taxonomy: "CPTY-REF-LEI", remediation: "cancel-correct",
            must_cite: "RTS22-Fld-Cpty", must_not_exceed: reporting.write },
  fail_if: submitted_without_dual_control  // right answer, wrong path = FAIL
}
```

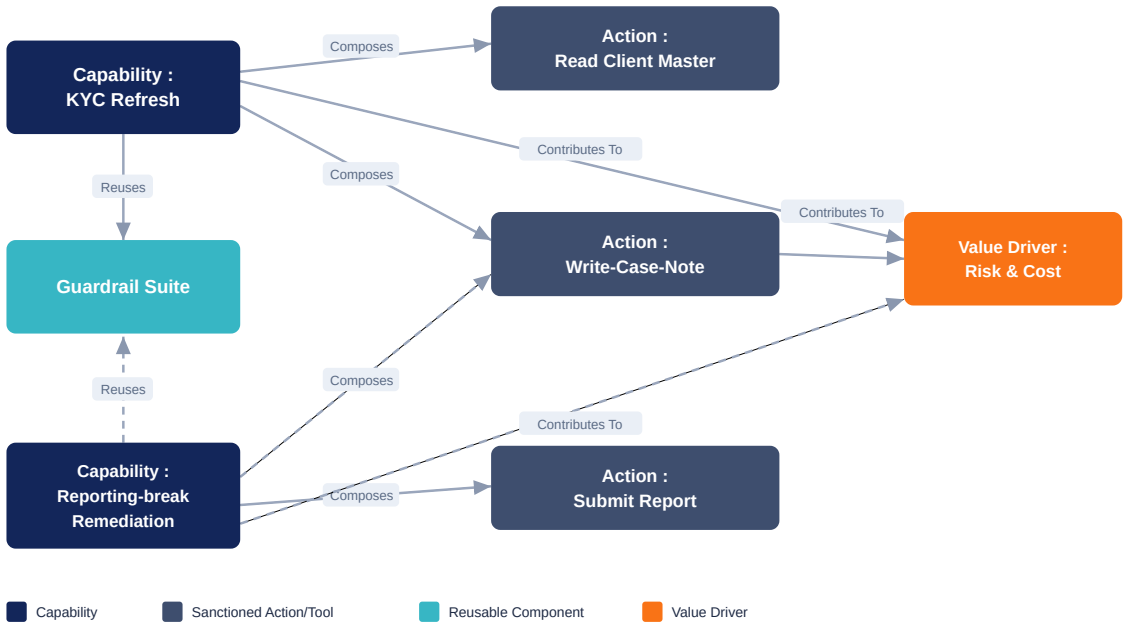
These run in a **simulation sandbox** with a mocked trade repository — including adversarial and edge breaks — before release and continuously after, inside an evaluator–optimizer loop. On economics, cost is governed per completed remediation, not per token: the routine majority of breaks are remediated straight-through on small models under human sign-off, with a frontier model reserved for the ambiguous root-cause tail, and progressive disclosure holding context — and therefore cost — down. The reported figure is cost per remediated break, typically a small fraction of the analyst time it replaces, with the reliability of each step measured, not assumed.

> **What good looks like**
>
> The routine majority of breaks are remediated straight-through under dual-control sign-off, inside the T+1 window; every remediation is replayable and cites the exact rule; nothing is submitted without a passing deterministic validation and an authorization gate; and the four skills are reused by adjacent capabilities — KYC refresh, for instance, inherits the evidence and reference-data skills without rebuilding them.

5. From silos to a capability map: the value graph for agents

The most expensive failure is not technical; it is that every team builds its own agent, its own integrations and its own controls over the same systems. The remedy is to make the enterprise's sanctioned capabilities a shared, machine-readable graph. In an agentic world this graph is not a slide — it is the map the **action catalog** and the orchestrator both read.

Figure 7 — Capability graph: new agents compose sanctioned actions instead of rewiring integration



Dashed edges are what the discovery agent surfaces: Remediation reuses the Write-Case-Note action and the eval/guardrail suite already sanctioned for KYC Refresh — no duplicate integration, no duplicate controls.

The schema is small enough to start this quarter.

```
NODES: Capability | Action(sanctioned tool) | SystemOfRecord | Control
       | ValueDriver | AgentComponent{Guardrail|EvalSuite|Memory|Agent}

EDGES: COMPOSES | READS/WRITES | GOVERNED_BY | CONTRIBUTES_TO | REUSES
```

Priority becomes a queryable number, scoring each capability on value (key performance indicator [KPI] impact, reach, risk reduction) against feasibility (data readiness, control burden, and — decisively — how much can be reused). **And the discovery agent is a traversal** that finds high-value capabilities already composable from sanctioned actions:

```
MATCH (c:Capability)-[:CONTRIBUTES_TO]->(:ValueDriver)
WHERE c.priority > 70
OPTIONAL MATCH (c)-[:COMPOSES|REUSES]->(a:AgentComponent)
RETURN c.name, c.priority, collect(a.name) AS reusable
ORDER BY c.priority DESC
```

In Figure 7 this is the Reporting-break Remediation case: the traversal surfaces that it can reuse the Write-Case-Note action and the eval/guardrail suite already sanctioned for KYC Refresh — no duplicate integration, no duplicate controls, no duplicate model-risk validation. Silos collapse because sanctioned capability becomes common, discoverable property rather than tribal knowledge.

6. Execution: a maturity model and a first-90-day path

Autonomy is earned, not granted. The right sequence raises autonomy only as the assurance to support it is built. Most institutions can locate themselves on this curve, and the goal is to move one level deliberately — never to grant level 5 to a system that has not passed level 3's gate.

Figure 8 — the autonomy staircase: raise the slider only as verification allows. After Karpathy's "autonomy slider" and "Iron Man suit" framing of partial autonomy.

Level	What it looks like	The gate to the next level
1 · Assisted	Copilots advise; humans act.	A harness and telemetry exist for the use case.
2 · Tool-using	Agents read and draft; humans execute.	Per-action authorization and delegated authority in place.
3 · Supervised-autonomous	Agents act on low-consequence steps; humans approve the rest.	Trajectory capture, replay and trajectory-level evals.
4 · Multi-agent	Agents compose via typed hand-offs under a supervisor.	Budgets, circuit breakers and semantic observability.
5 · Compounding	New agents compose sanctioned actions from the graph.	Reuse and value-graph governance are the default.

Ninety days is enough to prove the model on one real workflow — a bounded, high-value agentic use case such as reporting-break remediation or KYC refresh, not a demo.

Window	Focus	Exit criteria
Days 0–30 — Spine & control	Build the deterministic spine and action gateway for one workflow; wire delegated authority and the trajectory store.	One agent acting on low-consequence steps behind human approval, fully attributable and replayable.
Days 31–60 — Assurance & economics	Trajectory-level evals and a simulation sandbox; per-step routing and trajectory budgets; semantic observability.	Validated process-level evals; cost per completed task under budget; drift monitoring live.
Days 61–90 — Compounding	Publish the action catalog into the value graph; enable reuse detection; the discovery agent proposes the next two capabilities.	A reuse-aware capability map and a second agent built mostly from sanctioned actions.

7. From financial services to every regulated sector

A closing observation in a different register from the engineering above — commercial rather than technical, and included because it changes the business case for doing the hard work first.

The blueprint is financial-services-first by design, not by limitation. Because this sector imposes the most demanding overlay — the strictest controls, the hardest audit, the tightest deadlines — a platform hardened here extends to lighter-regulated sectors by **re-parameterization, not rebuild.** The engine is invariant; only the overlay changes.

What stays the same. The challenge catalog, the reference architecture (the deterministic spine, the control and assurance planes), the agent-identity and per-action authorization model, the evaluation and observability discipline, the capability graph and the maturity model are all sector-independent. What is swapped: only the overlay — the regulatory and control layer, the value ontology (which outcomes matter), the data domains and systems of record, and the specific action catalog.

Sector	The overlay that changes	Representative agentic use case	What carries over unchanged
Insurance	Fair-claims-handling and conduct rules; state / prudential regulation.	Claims adjudication; subrogation; underwriting assistance.	The entire spine, authorization model and assurance plane.
Healthcare & life sciences	the Health Insurance Portability and Accountability Act (HIPAA); clinical safety; the Food and Drug Administration (FDA) / GxP.	Prior-authorization and clinical-documentation support; safety-case triage.	Deterministic control; per-action authorization; trajectory audit.
Public sector	Due process; records and transparency mandates.	Casework and benefits adjudication; correspondence handling.	Agent identity; dual control; reproducibility.
Legal & professional services	Privilege; conflicts; duty of care.	Contract and disclosure review; diligence.	Barrier-scoped memory; typed hand-offs; evidence generation.
Telco & utilities	Sector regulation; critical-infrastructure rules.	Network-incident and billing-exception remediation.	Orchestration, budgets, circuit breakers and observability.
Manufacturing & supply chain	Safety and trade-compliance rules.	Exception, quality-deviation and customs remediation.	The capability graph and the reuse model.

The lesson is counter-intuitive but robust: the sector that looks hardest to enter is the one worth entering first. Solve agentic AI where the controls are unforgiving, and every other sector becomes a lighter version of a problem already solved — the same engine, a new overlay, at a fraction of the first build's cost.

8. Conclusion

The industry has convinced itself the agentic race is about model capability. In financial services it is not. The firms that win will be the ones that solved the unglamorous plane beneath the model — authorizing actions rather than data, giving agents identity, owning control flow deterministically, reproducing whole decision paths, budgeting trajectories, and turning integrations into a shared, governed action catalog. Those are the problems that keep agents out of production today, and they are solvable now. Solve them in the hardest regulatory environment there is, and the same spine, control plane and assurance plane extend to the next sector by swapping the domain overlay — not by rebuilding. That is how AI value is actually unlocked: not one clever agent, but an institution where the next agent is safer, cheaper and faster to build than the last.

8.1 Five questions worth asking of your own estate

A useful test of production-readiness is whether the following can be answered today, without a project to find out:

- Can you replay, step by step, the last decision your most autonomous agent made — and show a validator why it chose that path, not merely what it output?
- If an agent took a wrong action tonight, on whose authority did it act, and how quickly could that authority be revoked — in seconds, or at the next token refresh?
- What is your cost per completed task for your busiest agent, and how much does it vary from one run to the next?
- Would your evaluation fail a correct answer reached by a non-compliant path — or does it only grade the answer?
- How many teams have independently rebuilt the same integration, entitlement check and guardrail over the same system of record?

Few institutions can answer all five cleanly today. The gap between the confident answers and the uncertain ones is a fair map of the work ahead.

8.2 How Wissen engages

Our engagement model mirrors this paper. We start with a short assessment of a target workflow against the challenge catalog and the reference architecture — what exists, what is missing, where the risk concentrates. We then stand up one production-grade agent on a real workflow, typically within a first-90-days horizon, built on the deterministic spine with its control and assurance planes in place from day one. From there the work compounds: the governed action catalog and capability graph let each subsequent agent reuse what the first one proved, so the second is cheaper and faster than the first. The aim is not a demonstration; it is a governed platform your teams can build on and your second line can sign off.

9. Sources and a note on originality

The analysis, architecture, worked example and diagrams in this paper are original to the author. They synthesize and apply concepts that are, individually, established in the public domain — the deterministic control plane, orchestrator–worker decomposition, per-action authorization, agent identity, Agent Skills and the Model Context Protocol — to the specific problem of transaction-reporting break remediation at a regulated financial institution. The paper's contribution is that synthesis and application, not a claim to have invented the underlying patterns.

Selected references and influences. This paper is in conversation with a body of public work, which it applies rather than reproduces. On the reliability and autonomy of agents: Andrej Karpathy — "Software 3.0," the "decade of agents," the "march of nines," and the autonomy slider / "Iron Man suit" framing of partial autonomy. On agent design patterns: Anthropic's Building Effective Agents (workflows versus agents; orchestrator–workers; evaluator–optimizer), its Agent Skills open standard and context engineering guidance, and the Model Context Protocol. On the canonical agent decomposition (planning, memory, tool use): Lilian Weng. On agent security: Simon Willison's "lethal trifecta." On evaluation as a discipline: Chip Huyen and Shreya Shankar ("who validates the validators?"). For regulatory grounding: U.S. Federal Reserve / OCC SR 11-7, the PRA's SS1/23, and the EU AI Act's high-risk obligations taking effect through 2026. Specific factual claims are cited in the end notes below.

10. References

1. On the pilot-to-production gap, 2026 surveys converge on roughly one in ten agent pilots reaching production — Forrester / Anaconda 2026 and WRITER / Workplace Intelligence 2026 (n=2,400) report that about 88% never graduate — with evaluation coverage (~64%), governance (~57%), data quality and security cited as the leading blockers rather than model capability (Gartner 2026 Hype Cycle for Agentic AI; S&P Global Market Intelligence / McKinsey 2026, which find banking and insurance leading sectors on production adoption).

2. Rule-based AML transaction monitoring commonly produces false-positive rates above 95% (some estimates near 98%), and roughly two-thirds of financial-crime-compliance cost sits in KYC / customer due diligence — LexisNexis True Cost of Compliance (2024); Thomson Reuters (2025); AML-technology research (arXiv, 2025–26).

3. Global penalties for AML, KYC, sanctions and customer-due-diligence failings totaled approximately USD 3.8 billion in 2025 (down from USD 4.6 billion in 2024), the single largest about USD 985 million — Fenergo annual enforcement findings (January 2026).

4. U.S. Federal Reserve SR 11-7 / OCC Bulletin 2011-12, "Guidance on Model Risk Management" (2011); in the UK, PRA Supervisory Statement SS1/23 on model risk management principles.

5. On agent authorization, see 2026 work on invocation-bound and attenuated capability tokens and formal results that time-to-live revocation is insufficient at agent execution speed — e.g. preprints on authorization propagation in multi-agent systems (arXiv, 2026) and the OpenID Foundation agentic-identity whitepaper (2025).

6. The "Know-Your-Agent" framing — verifiable agent identities linked to a legal entity — appears in 2026 International Monetary Fund (IMF) and World Economic Forum analyses of agentic AI in payments and finance.

7. On retrieval at scale, 2026 analyses show approximate-nearest-neighbor recall degrades as a corpus grows and under metadata / entitlement filtering, regardless of engine — e.g. filtered-ANN system benchmarking (arXiv, 2026) and practitioner analyses of HNSW (Hierarchical Navigable Small World) recall–latency behavior at scale. Hybrid retrieval that pre-selects candidates via metadata / graph filters is the reliable mitigation.

8. Agent Skills — folders of instructions, scripts and resources loaded by progressive disclosure — were introduced by Anthropic and released as an open standard (December 2025), since adopted across multiple model providers and agent runtimes. See Anthropic, “Equipping agents for the real world with Agent Skills”; the Agent Skills specification at agentskills.io.

9. Model Context Protocol (MCP): an open standard for connecting agents to external tools and data, introduced by Anthropic (2024).

About Wissen

Founded in 2000 in the US, Wissen is an esteemed information-technology company headquartered in Bangalore, India. With a global presence spanning offices in the US, India, UK, Australia, Mexico and Canada, we offer end-to-end solutions to companies across sectors such as Banking and Financial Services, Telecom, Healthcare, Manufacturing and Energy. Leveraging state-of-the-art infrastructure and development facilities worldwide, we have successfully delivered over $1 billion worth of projects for more than 25 Fortune 500 companies. With a highly skilled workforce of 5,500+ professionals across the Wissen Group, we combine deep domain expertise with engineering rigor to deliver measurable outcomes for the world's most demanding institutions.

Services We Offer

ADVISE

- Regulatory, Compliance & Risk Advisory
- AI Governance & Enablement
- Target Operating Model & Cost Transparency

BUILD

- Application & Platform Engineering
- Data Engineering & Analytics
- AI/ML & Agentic Automation
- Cloud & DevSecOps
- Product Design
- QA & Test Automation

RUN

- Managed Services & KPO
- Regulatory Reporting as a Managed Service
- Infrastructure & SRE

✉ contact@wissen.com

WISSEN TECHNOLOGY

DELIVERING 2X IMPACT