Agentic Architectures in Finance

A Governance-First Companion Handbook
to 10 LangGraph Colab Notebooks

Alejandro Reynoso

Chief Scientist, DEFI Capital Research
External Lecturer and Research Fellow, Judge Business School, University of Cambridge

February 19, 2026

Contents

Introduction I: What This Handbook Is (and Is Not)

This handbook is a companion to a specific kind of work: the construction of agentic systems that operate inside financial organizations under real constraints, real accountability, and real reputational and regulatory risk. It is not a collection of clever prompts. It is not a catalog of "AI tricks." It is not a pitch deck for autonomy. It is a governance-first operating manual for building, running, and reviewing LangGraph-based workflows that behave like disciplined teams: they receive messy requests, separate what is known from what is assumed, route decisions through explicit gates, stop when they hit a bound, and leave behind artifacts that let a human committee understand what happened and why.

The core promise is simple. In most firms, the hard part of using AI is not generating text. The hard part is building a system that can be inspected after the fact. Committees do not approve outputs; they approve processes. They want to know where information came from, what rules were applied, what was deterministic, what was probabilistic, what was inferred, and what was escalated to a human. They want to know whether the workflow can be reproduced tomorrow, under audit, when the same request arrives with slightly different wording. They want to know whether the system fails safely. This handbook exists because "a model answered" is not a control, and "it usually works" is not a governance standard. The objective is reviewability: a system that produces outputs the organization can stand behind because it can show its work.

This handbook is therefore written as if the author is a senior analyst inside a firm presenting to top management, to the risk committee, and ultimately to the board. Each chapter takes an architecture that resembles a real internal workflow and explains it in operational language: what problem the workflow solves, how it decomposes the work into nodes and responsibilities, where the firm hard-codes constraints, where it uses model reasoning, where human judgment is required, and what artifacts are produced to ensure accountability. The tone is deliberate. The intent is not to dazzle. The intent is to make it possible for a committee to ask hard questions and receive concrete answers.

The book is also a practical bridge between two worlds that are often separated. On the one hand, there is the world of code: notebooks, state machines, conditional edges, error handling, determinism, and logs. On the other hand, there is the world of financial decision-making: incomplete information, ambiguous objectives, timeline pressure, institutional constraints, and layered accountability. Too many AI write-ups live only in one world. Purely technical descriptions do not translate into governance decisions; purely conceptual narratives cannot be executed or tested. This handbook insists on both. Each architecture is paired to a Colab notebook and is explained down to the level of "Cell 1 does this, Cell 2 does that," because the committee must understand which parts of the workflow are infrastructure, which parts are controls, which parts are model calls, and which parts are deliverables.

The guiding idea is that agentic systems are not assistants; they are processes. In a financial organization, a process is something that must be bounded, documented, and reviewable. It has a scope, a set of inputs, a set of outputs, and explicit rules that determine when the process should stop or escalate. It also has a lifecycle:

development, testing, pilot, monitoring, and periodic review. This handbook focuses on the architecture layer that makes that lifecycle viable. The architectures in the companion notebooks are constructed using LangGraph because it forces explicitness: state transitions are defined; routes are conditional and inspectable; loops can be bounded; termination is explicit; and the graph itself can be rendered and stored as an artifact. That structure is not aesthetic. It is governance.

The "common denominator" across the ten chapters is an operating discipline that is intentionally conservative. Each notebook uses an explicit TypedDict state. This is not a Python style preference. It is a way to prevent invisible memory. If the system must carry facts, it must carry them in fields that can be reviewed. If the system must track assumptions, it must record them explicitly. If the system must ask follow-up questions, it must store what was asked and what was answered. If the system must route, it must store the route. The state is the system's memory, and making it explicit is a precondition for accountability. This is why this handbook repeatedly returns to the same pattern: define the state, define the nodes, define the gates, define the loop bounds, define the termination reasons, and export the artifacts.

This handbook also takes a particular stance on "what the system is allowed to do." In professional settings, the most dangerous failure mode is not that the model is wrong. The most dangerous failure mode is that the model appears right. Overconfidence, plausible narratives, and the quiet conversion of uncertainty into fluent text are the signature risks of LLM usage. A governance-first architecture therefore treats model output as a draft inside a controlled workflow, not as a conclusion. The system's job is often to produce a structured packet for a human decision-maker, not to replace the decision-maker. This is why the architectures emphasize triage packets, evidence-gap registers, decision tickets, escalation triggers, and refusal paths. A safe system has the ability to say: we do not have what we need, and we are stopping here.

So what is this handbook, concretely. It is a small, operational companion to ten Colab notebooks. Each chapter corresponds to one notebook and one architecture pattern. The patterns are chosen because they are common in financial organizations: triage of requests, suitability boundaries, credit memo drafting with missing evidence, hypothesis testing via backtest wrappers, execution tactics under liquidity regimes, portfolio rebalancing with competing risk and cost objectives, investment banking pitchbook production, M&A diligence Q&A over document corpora, treasury monitoring and covenant triggers, and multi-desk research synthesis with red-teaming. These are not toy examples; they are stylized versions of workflows that exist in real firms. The notebooks use synthetic or controlled inputs, but the architecture patterns are intended to transfer to real environments. The chapters explain the patterns at the level that matters: governance.

Each chapter is designed to be read by two kinds of readers. The first is the executive reader: someone who needs to understand why the firm would invest in such an architecture, what risk it reduces, what deliverable it produces, and what controls are non-negotiable. The second is the builder and reviewer: someone who needs to map that narrative to the actual notebook cells and confirm that the code implements the claimed controls. This dual audience is not a compromise. It is the point. A governance-first system cannot be "understood" only by engineers and "approved" only by executives. The process has to be legible across the institution.

Because of that, each chapter in this handbook follows a fixed structure. First, it provides a board-facing

narrative of the enterprise problem the architecture solves and why naive LLM usage is unacceptable. Second, it explains the technology in institutional language: what a node is, what an agent is, where hard-coded logic lives, where model reasoning is used, and where humans intervene. Third, it lists the governance controls: bounded loops, hard gates, termination reasons, artifact bundles, and failure mode handling. Fourth, it provides a cell-by-cell walkthrough of the paired notebook, so that reviewers can trace how the process is implemented. Finally, it ends with a conclusion that is not promotional but developmental: what should be improved next, what should be measured, what additional controls may be needed before broader deployment, and what scope restrictions are prudent.

This handbook is not a replacement for documentation, policy, or compliance review. It is not a substitute for model risk management. It is not a legal interpretation of regulations, suitability rules, fiduciary obligations, or internal policy. When a workflow intersects with those domains, the handbook names that intersection explicitly and treats it as a governance boundary. A critical distinction is made throughout: the system can help structure information and draft internal deliverables, but the firm must define what constitutes advice, what constitutes a recommendation, what constitutes solicitation, and what constitutes an action. Those definitions are firm- and jurisdiction-specific and must be reviewed by qualified professionals. This handbook therefore assumes a conservative stance: outputs are labeled as drafts; verification status is not assumed; escalation to human review is explicit; and refusal/redirect paths exist when the request is out of scope.

Likewise, this handbook is not a promise of performance. It does not claim that the architectures eliminate human error or reduce all risk. It claims something narrower and more defensible: these architectures reduce certain classes of failure by making process explicit, bounded, and reviewable. They create an environment where failure modes can be seen, measured, and improved rather than hidden behind fluent prose. They enable controlled pilots, because the organization can define success and failure in terms of routes taken, stop reasons, and artifact completeness. They enable audits, because the system leaves behind a structured record. They enable governance, because decision-makers can demand that the system meet minimum standards before it is used in sensitive settings.

This handbook is also not a monograph on LangGraph, nor is it a tutorial on Python. The notebooks are executable, and the chapter cell walk-throughs explain the intent of each cell, but the purpose is not to teach programming for its own sake. The purpose is to teach institutional architecture: how to build AI workflows that behave like controlled processes. The technical details are included only insofar as they support that objective. When the handbook describes a Mermaid graph, it is not to impress with diagrams; it is to explain that the routing logic is explicit and can be reviewed as a graph specification artifact. When the handbook describes a TypedDict state, it is not to show typing; it is to explain that information is tracked explicitly and can be inspected. When the handbook describes bounded loops, it is not to demonstrate control flow; it is to explain that the process cannot run away and cannot quietly "keep trying" until it fabricates a confident answer.

An important implication follows. Because the architecture is a process, the primary deliverable is not always a piece of prose. In many chapters, the deliverable is a structured packet: a triage packet, a credit memo draft plus gap register, a backtest decision ticket, an execution plan with regime rationale and stop reasons, a

rebalance proposal with committee dissent, a pitchbook section set with consistency checks, a diligence Q&A pack with provenance, a treasury alert report with trigger logic, a research synthesis with red-team objections. Those deliverables are what the committee should evaluate. Text is a container; governance is in the process that produced it.

This is also why the notebooks generate artifacts every run. A board does not need to read code, but it does need to know that the firm can reconstruct what the system did. The run manifest plays the role of an audit header: it records configuration, environment, run identifiers, and other reproducibility signals. The graph specification makes the routing logic reviewable: what nodes exist, how they connect, and under what conditions the workflow transitions. The final state is the ledger of the run: what inputs were received, what intermediate judgments were made, what missing items were identified, what loops executed, what termination reason was triggered, and what final decision was produced. In a governance-first system, these artifacts are not optional extras. They are the only reason the system is institutionally usable.

The handbook is therefore designed to be used in a particular rhythm. A reader begins with a chapter narrative to understand why the pattern exists and what it claims to control. Then the reader runs the paired notebook to observe the pattern in motion. Then the reader inspects the artifacts to confirm that what happened matches the narrative. Then the reader modifies scenario parameters to stress the workflow: increase ambiguity, increase missing information, increase regime volatility, tighten constraints, and observe whether the workflow remains bounded and reviewable. This rhythm is not academic; it is the foundation for internal model risk review. A committee can only approve a process that can be tested, and it can only be tested if it is observable.

A word on "human intervention," because it is a recurring theme and it must be treated with precision. The architectures in this handbook are not designed to remove people. They are designed to make people more effective by structuring their work and preventing predictable failure. The human role is not an afterthought; it is a node in the process. In some cases, the human is the final approver. In other cases, the human is the source of missing information. In high-stakes cases, the human is the only permissible decision-maker, and the system's job is to triage and package. The design principle is that the workflow should make it difficult to accidentally skip human review when it is required. That is why the systems include hard gates and early termination. A safe system fails closed.

If this handbook insists on transparency and governance, it is because financial organizations do not merely operate under uncertainty; they operate under scrutiny. A process that cannot be explained becomes a liability, no matter how "smart" it appears. The board's concern is not whether the model can write a memo; it is whether the organization can defend the memo's production process. The committee's concern is not whether the system is impressive; it is whether it is safe. The risk function's concern is not whether the system can generate an answer; it is whether the system can be monitored and controlled. This handbook is written to satisfy those concerns.

What this handbook is not, finally, is a claim that agentic systems are inevitable or always desirable. Sometimes the best governance is simply not to automate. There are decisions that should remain human-only because the consequences are high and the inputs are too contextual, too sensitive, or too difficult to verify. This

handbook explicitly supports that stance by including refusal paths, redirect paths, and termination logic. A mature architecture does not try to do everything. It tries to do a bounded set of tasks well, with controls, and it stops.

This introduction therefore sets the contract for the rest of the book. If you are looking for inspirational narratives about AI "replacing analysts," you will not find them here. If you are looking for architectures that resemble actual enterprise workflows and can be defended in front of a committee, you are in the right place. The chapters that follow are small by design and strict by design. They are meant to be run, inspected, and debated. They are meant to produce disagreement in the right place: not about whether the model's prose was eloquent, but about whether the process was sufficiently governed, sufficiently transparent, and sufficiently bounded for the organization's risk appetite.

What this handbook is

- A governance-first companion to ten LangGraph notebooks implementing auditable agentic workflows in finance.
- A board-facing explanation of what each architecture solves, how it works, where humans intervene, and what artifacts make it reviewable.
- A practical bridge between code-level implementation (cells, nodes, state) and institution-level approval (controls, escalation, auditability).

What this handbook is not

- Not financial advice, not compliance guidance, not legal interpretation, not a substitute for qualified professional judgment.
- Not a catalog of prompts or a claim of autonomy; these are controlled processes with hard gates and escalation.
- Not a performance guarantee; the claim is bounded: transparency, governance, and safer failure modes through explicit structure.

Chapter 1

Personal Finance Triage and Missing-Info Discovery

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_1.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

In most financial organizations, the fastest-growing source of operational risk is not a failed trade, a breached covenant, or a broken model. It is the everyday request for "quick guidance" that arrives without a case file. A relationship manager forwards a client message asking whether they should refinance. A senior executive asks if a family trust should shift from cash to equities "before rates move." An employee asks whether their insurance coverage is adequate. The request looks small, conversational, and urgent. It is also incomplete. The real risk is not the question itself; it is the gap between what the stakeholder thinks they asked and what the organization would need to know to respond responsibly.

That gap is structural. Stakeholders compress context because they are busy, because they assume the firm "already knows," or because they do not know which facts are decision-critical. Even within the same institution, different desks and functions interpret the same question differently. "Is this rate good?" might mean "is it competitive in the market," "is it suitable for my horizon," "is it consistent with my liquidity constraints," or "does it create refinancing risk under income volatility." A naive chatbot-style interaction tries to be helpful by completing the picture. It fills in missing facts with plausible assumptions, converts uncertainty into fluent prose, and produces an answer that sounds like advice. The organization then inherits a liability: it has created a recommendation-like output without a controlled intake process, without a documented evidence base, and without a clear human sign-off boundary.

Notebook 1 (N1) exists to solve a narrower and more defensible problem than "give advice." It operationalizes a triage and missing-information discovery workflow. The purpose is to convert an ambiguous, incomplete request into an intake-ready triage packet that a qualified human can review. In this architecture, the model is not treated as an advisor. It is treated as a drafting component inside a governed process whose primary product is a structured case file: what was provided, what is missing, what must be clarified, and what the appropriate escalation route is.

The enterprise scenario is recurring and easy to recognize. Someone inside the firm is asked to respond quickly, but they do not have the minimum inputs that would be required under professional standards of care. The temptation is to respond anyway, because "doing something" feels better than asking questions, and because stakeholders often interpret follow-up questions as delay. The triage architecture resolves this tension

by making follow-up questions the default and by bounding that questioning so the process cannot drift. It behaves like a disciplined analyst team with a clear operating mandate: first, determine whether the request is suitable for automated intake; second, identify critical missing information; third, ask targeted follow-ups within a fixed budget of attempts; fourth, package a summary for human review; and fifth, refuse or redirect when the request is unsuitable or unsafe.

The deliverable is therefore not "personal finance advice." The deliverable is a triage packet. In board language, this is an internal control object. It contains a normalized restatement of the stakeholder's ask, a registry of facts explicitly provided, a list of assumptions that were proposed but not verified, an open-items list of missing critical fields, a set of recommended follow-up questions, and a routing recommendation such as HUMAN_REVIEW, REFUSE, or REDIRECT. It also includes explicit stop reasons: whether the process stopped because it reached a minimum completeness threshold, because it exhausted its retry budget, or because it encountered an unsuitability trigger.

This is exactly the point where governance becomes tangible. A committee does not approve "AI output." It approves an operational process and the controls that bound it. The missing-information discovery loop is the central distinctive element in N1. It is designed as a bounded retry loop with explicit counters. Each iteration produces a set of questions tied to specific missing items, captures responses, updates the state, and re-evaluates completeness. If the missing items cannot be resolved within the bounded number of attempts, the system stops and routes to human review rather than fabricating completeness. The loop therefore serves two governance functions at once: it improves the intake packet by collecting missing data, and it prevents uncontrolled interaction that could produce overconfident or unreviewable outputs.

The failure modes that matter to the board are predictable. Overconfidence under missing information is the signature risk. A fluent answer that quietly assumes time horizon, liquidity needs, tax constraints, or existing exposures can cause harm even if the arithmetic is correct. Bad question selection is the next risk: asking irrelevant follow-ups wastes stakeholder patience and can create the perception that the firm is disorganized. False completeness is more subtle: the system may declare "enough information" when a critical field is still unknown, leading downstream reviewers to treat an incomplete packet as complete. Loop exhaustion without escalation clarity is another governance hazard: the system keeps asking until it hits its cap, but if the stop reason is not explicit, users may misinterpret "we stopped" as "we finished." Finally, stakeholder misinterpretation is a communications risk: even if the triage packet is intended for internal review, stakeholders may treat any system-produced language as an approval. Labeling and gates must therefore be part of the process design, not an afterthought.

N1 addresses these failure modes with process, not with persuasion. It forces the system to maintain a clear separation between facts, assumptions, and open items. It forces bounded loops with explicit termination. It forces a routing recommendation that includes a human intervention point. It leaves behind artifacts that allow a reviewer to reconstruct exactly what happened: what information was provided, what was missing, what questions were asked, what responses were received, and why the workflow stopped.

To explain the architecture in the language of a firm, imagine a small internal intake desk that sits between

stakeholders and licensed professionals. The desk has a strict script. It does not provide advice. It converts conversation into a structured intake file. It records what it heard, identifies what it needs, asks for the missing items, and then forwards the file to a human. If the request is unsafe—if it seeks a guaranteed return, asks for advice outside scope, or lacks enough context to proceed—the desk refuses or redirects. The N1 workflow is that desk, encoded as a graph.

From a board perspective, the key question is: what should we approve? The decision packet embedded in this chapter is intentionally conservative. First, approve a pilot for triage-only intake where outputs are explicitly labeled as internal intake packets and where every packet requires mandatory human sign-off before any downstream action or communication. Second, restrict categories that are high-stakes or inherently suitability-sensitive to human-only handling; the system may still triage and collect missing data, but it must route to HUMAN_REVIEW early and consistently. Third, require metrics and audit artifacts as release criteria: the workflow must produce run_manifest.json, graph_spec.json, and final_state.json for every run, and the organization must be able to show, for a sample of cases, that the routing and stop reasons were appropriate.

The pilot recommendation also includes explicit prohibitions. The workflow must not produce personalized recommendations, must not present numerical projections as certain when inputs are missing, and must not allow "silent proceed" when completeness thresholds are not met. The system may draft language, but the institution must retain the human gate as the decision-maker. Where compliance boundaries exist—what counts as advice, what disclosures are required, what data retention policies apply—those are open items for compliance review and must be resolved before any client-facing deployment.

What does "success" look like for N1? It is not measured by whether the system produces impressive prose. It is measured by whether the organization reduces unstructured, undocumented responses and replaces them with a reviewable intake process. A successful run produces a triage packet that a professional can act on quickly because it is structured, explicit about unknowns, and honest about what was not provided. It also produces artifacts that allow internal audit and model risk functions to test the workflow: how often it escalates, how often it exhausts loops, how often it identifies missing critical fields, and whether its question selection is improving.

In short, Notebook 1 formalizes a simple institutional principle: in finance, you do not answer a question until you know what question you are answering. N1 encodes that principle as a governed process. It makes the gap between stakeholder intent and institutional requirements explicit, it asks for what is missing, it stops when it must, and it hands the decision to a human when the stakes or uncertainty demand it.

The governance rationale for insisting on artifacts is practical, not philosophical. When a complaint arises, when a regulator asks how a communication was produced, or when a supervisor questions why a case was escalated, the organization must be able to reconstruct the process. The run manifest provides a run identifier, a configuration fingerprint, and an environment snapshot, so that the workflow can be reproduced. The graph specification provides the approved routing topology: which nodes exist, which decisions are hard-coded, and which transitions are allowed. The final state is the case ledger: it shows the evolving intake file across the

loop—what the system extracted as facts, what it labeled as missing, what questions it asked, what answers were captured, and what termination reason concluded the workflow. Without these artifacts, the firm can only say "the model responded." With them, the firm can say "this was the controlled process, these were the inputs, these were the steps, and this was the explicit reason we escalated."

The triage packet itself should be treated as a standard document type. It should have a stable schema so downstream reviewers can operate quickly. A well-designed schema includes at least four compartments. The first compartment is the normalized request: one sentence that restates what decision is being contemplated and on what timeline. The second compartment is the facts provided: the amounts, rates, dates, incomes, debts, account types, and constraints that the stakeholder explicitly supplied. The third compartment is the open-items list: the minimum missing facts that prevent responsible analysis, prioritized by decision-criticality. The fourth compartment is the escalation recommendation: whether a human professional should take over immediately, whether further information should be collected, or whether the request should be refused or redirected because it is unsuitable.

For the board, it is useful to think of this workflow as a safety layer that sits upstream of every "advice-like" interaction, whether the firm is serving clients or supporting employees. The same control logic applies across these contexts: do not allow an LLM to create the appearance of professional judgment when the case file is incomplete. The architecture therefore institutionalizes a cultural shift. It makes it normal to ask the "boring" questions first, and it makes it measurable. How often do requests arrive missing a time horizon? How often is liquidity the hidden constraint? How often do stakeholders omit existing debt covenants or insurance exclusions? The workflow does not merely respond; it creates a dataset of gaps that the organization can use to improve intake forms, training, and client communication.

Finally, the architecture creates a place for human intervention. Human review is not a patch applied at the end. It is a designed gate with a trigger. The system can proceed through the discovery loop, but only a human can approve downstream action, and the routing decision must be explicit. In the pilot phase, the firm should require that reviewers sign off on the triage packet as complete enough to analyze, or mark it as incomplete and return it for additional intake. Over time, this review feedback becomes a governance asset: it can be used to improve question selection, adjust completeness thresholds, and calibrate escalation rules without changing the fundamental safety posture.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

The N1 workflow is implemented as a LangGraph state machine, and the architecture explanation should be read as a description of a controlled process rather than an "intelligent assistant." LangGraph forces the designer to declare, in advance, what the system can do: which nodes exist, what each node is responsible for, and which edges connect them. An edge is not a vague intention; it is an allowed transition that may be unconditional or conditional. A conditional edge is the mechanism that turns policy into routing: if a completeness gate passes, the graph can proceed to packaging; if it fails and retries remain, the graph loops into another follow-up cycle; if it fails and retries are exhausted, the graph terminates into an escalation path.

The system's memory is the state. In this notebook, the state is defined as an explicit TypedDict, which matters because it prevents hidden memory and makes review possible. State fields represent the incoming request, its normalized interpretation, a set of extracted facts, a set of missing critical items, the follow-up questions generated for those items, the follow-up responses collected from the user, counters that track how many discovery iterations have occurred, and final fields such as termination reason and routing decision. A reviewer does not need to "remember" what happened; they can inspect final_state.json and see the full case ledger.

A node is the unit of work that reads part of the state and writes updates back to the state. In this project, nodes are wrapped in an AgentNode abstraction to enforce single responsibility and consistent logging. Practically, that means each node has a name, an input contract, an output contract, and a structured trace of what it attempted to do. The word "agent" in this setting does not mean autonomy; it means a node whose internal step may include an LLM call for synthesis, classification, or drafting, but whose external behavior is bounded by the graph and the state schema.

The architecture deliberately separates hard-coded logic from model-driven reasoning. Hard-coded logic appears in the control surface: state initialization, loop counters, maximum retry constants, routing conditions, and termination reasons. These are deterministic and auditable. They are where governance lives in code. Model-driven reasoning is used only for bounded tasks that require language understanding: extracting candidate facts from free text, identifying which information is likely missing given the request type, drafting targeted follow-up questions, and summarizing the triage packet in a consistent format. Crucially, every model output is written into state as an object that can be reviewed. The system is not allowed to "just know" something; if it relies on a fact, that fact must be recorded as provided, inferred, or open.

A gate is an explicit decision function that evaluates state and decides which edge the graph takes next. In N1, the central gate is the completeness and safety gate. It answers: do we have the minimum critical fields to package a triage packet as intake-ready, or do we need another discovery iteration, or is the request unsuitable or unsafe such that we must refuse or redirect? The gate is implemented as deterministic logic over state fields, not as free-form model choice. This is important for reviewability. Committees can inspect the gate rules, test them against scenarios, and adjust them as policy without rewriting the entire workflow.

The bounded loop is the safety valve that prevents uncontrolled interaction. The loop repeats a sequence: update missing-items list, draft follow-up questions, capture responses, and re-evaluate. The loop is bounded by a maximum number of iterations. When the cap is reached, the system does not improvise; it terminates with an explicit stop reason such as MAX_RETRIES_REACHED and routes to HUMAN_REVIEW. This "fail closed" behavior is the difference between a governed process and an ungoverned chat. The bound also makes monitoring possible: loop exhaustion rate becomes a measurable indicator of intake quality and question selection quality.

Artifacts are the system's accountability layer. The run_manifest.json file provides reproducibility metadata: run identifiers, timestamps, configuration hashes, and environment fingerprints. graph_spec.json captures the graph topology and routing rules so that reviewers can see what the system was allowed to do. final_state.json

captures what the system actually did and what it believed at the end: extracted facts, missing items, questions asked, answers received, termination reason, and final routing decision. Together these artifacts let a risk committee ask and answer operational questions: Did the system escalate when it should have? Did it stop for the right reason? Did it misclassify missing items? Did it ask irrelevant questions? Did it ever proceed without minimum fields?

Human intervention is intentionally designed into the boundary conditions. The system's outputs are labeled as Not verified, and the routing decision is designed to land on HUMAN_REVIEW for any downstream action. Humans can intervene by correcting extracted facts, adding missing items, overriding completeness decisions, and approving or rejecting the triage packet as "intake-ready." In a mature deployment, human feedback would also be recorded as part of the state or as a linked review artifact, closing the governance loop.

In short, the basic architecture is a controlled intake machine: a graph of nodes that updates an explicit case state, uses model reasoning only inside bounded drafting tasks, enforces policy through deterministic gates and bounded loops, and exports artifacts that make the process inspectable and reproducible. The Mermaid rendering in the notebook is not cosmetic. It is a governance aid that turns the workflow into an object the committee can inspect: the nodes, the conditional branches, and the termination endpoints are visible as a diagram and also stored as a machine-readable graph specification. Likewise, key management is explicit: the model client is initialized only if `userdata.get("ANTHROPIC_API_KEY")` is present, and failures to load credentials are surfaced early as controlled errors rather than silent fallbacks. This prevents shadow mode surprises entirely.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

The board question for N1 is not "does it write well," but "does it make our process safer and more reviewable." If the organization adopts this intake workflow, the next step is to treat it as a controlled product with measurable operating performance and explicit change management. The first improvement is therefore governance hardening: define, in policy, what the triage packet is allowed to contain and what it is prohibited from containing. The packet should never read like a personalized recommendation. It should read like intake: facts, gaps, and a routing decision. The organization should standardize a header that labels the output as Not verified, internal-use by default, and subject to human approval. If the firm intends any client-facing use, disclosures and suitability boundaries are an open item for compliance review and must be resolved before deployment.

The second improvement is calibration of completeness thresholds. "Minimum critical fields" should not be static. They should be defined per request category and risk tier, and they should be reviewed periodically. A refinancing triage requires different minimum fields than an insurance coverage triage or a retirement withdrawal triage. The workflow should therefore maintain a category field and use category-specific completeness checklists. This can be implemented without changing the architecture: the gate reads a checklist from configuration and evaluates state accordingly. The board should require that these checklists be versioned, approved, and linked to run manifests so that any triage packet can be traced to the checklist

version that governed it.

The third improvement is question quality control. The most visible user experience risk in N1 is bad follow-up questions. The system should ask fewer questions, better. That requires two mechanisms. First, a prioritization policy that ranks missing items by decision-criticality and asks only the top items within the bounded loop. Second, a feedback loop from reviewers: when a human marks a question as irrelevant or confusing, that signal should be logged and used to refine the question templates or the missing-item classification prompt. This does not require training a new model; it requires disciplined logging and iterative prompt governance with regression tests.

The fourth improvement is explicit escalation hygiene. Loop exhaustion must not be ambiguous. When MAX_RETRIES_REACHED occurs, the system should produce a short escalation note that explains why further questioning did not resolve the gaps, what gaps remain, and what the recommended human next step is. Similarly, when a request is flagged as unsuitable or unsafe, the refusal or redirect should be framed as a policy outcome, not as a model opinion. The language should be standardized and approved. In many firms, this is the difference between a defensible refusal and a reputational incident.

The fifth improvement is production monitoring and metrics. The board should require a minimum monitoring dashboard before expanding scope. At least five metrics should be tracked. Routing accuracy: in a sampled set, did the workflow route to HUMAN_REVIEW when it should have, and did it avoid inappropriate "proceed" paths? Open-items rate: how many critical items remain open at termination, and how does that vary by category? Loop exhaustion rate: how often does the workflow hit the retry cap, which indicates either poor question selection or insufficient stakeholder responsiveness? Escalation frequency and refusal rate: are we escalating too little (unsafe optimism) or too much (uselessness)? Stability across scenarios: when inputs are paraphrased, does the workflow behave consistently, or does it route differently without justification? These metrics tie directly to model risk management: they show whether the system is predictable, conservative, and improving.

The sixth improvement is rigorous artifact discipline. The artifact bundle should be treated as mandatory output, not a development convenience. The organization should define a retention policy and a redaction policy, both as open items for compliance review, and then implement them consistently. In particular, prompts logs should be minimized and hashed; sensitive personal data should be avoided where possible; and any stored content should have a clear purpose. The board should also require periodic artifact audits: randomly select cases, reconstruct the run from the artifacts, and confirm that the workflow followed approved routing and that the triage packet matches the final state.

The seventh improvement is a conservative deployment plan that respects change management. Start with a pilot in an internal-only setting, such as a centralized service desk supporting relationship managers. Restrict the scope to triage-only outputs with mandatory human sign-off. Require reviewer training: reviewers must understand that the triage packet is not advice and that missing items are not optional. Establish a weekly review cadence for the first eight weeks to examine failure cases, question quality, and escalation patterns. Only after the workflow demonstrates stable routing and acceptable loop exhaustion should the firm consider

broader rollout.

Finally, the board should view N1 as a foundation layer rather than a standalone solution. The triage packet is an upstream artifact that can feed downstream governed workflows: suitability gates, credit memo drafting, research synthesis, or portfolio decision tickets. The value compounds when the organization standardizes the intake discipline across workflows. A firm that adopts N1 is adopting a principle: before judgment, structured intake; before conclusions, explicit gaps; before automation, bounded control. That principle is portable, measurable, and defensible.

The recommendation to the board is therefore specific. Approve N1 as a pilot intake control with strict guardrails: triage-only, internal by default, mandatory human sign-off, bounded loops, explicit termination reasons, and mandatory artifacts. Forbid recommendation-like language and forbid silent proceeding under missing critical fields. Require the monitoring metrics and the artifact audit program as conditions for scale. If these conditions are met, the organization will not merely have an AI tool; it will have an auditable process that reduces a common class of operational risk: the fast answer to the wrong question. As a final control, the firm should run red-team drills: intentionally incomplete requests, misleading numbers, and high-pressure language. The workflow should consistently slow down, request missing items, and escalate safely.

Chapter 2

Advice Suitability Boundary: Refuse / Redirect Paths

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_2.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

In every financial institution there is a moment when a request crosses an invisible line. A client asks a relationship manager, "Should I sell this position and buy that one?" An employee asks a benefits team, "How much should I contribute to my retirement plan?" A senior executive forwards a message and says, "Give me a quick answer so I can reply." The words are casual, but the implication is not: the institution is being asked to provide something that can be interpreted as individualized advice. The operational risk is not only that the answer might be wrong. The risk is that the organization delivers an answer without first determining whether it is allowed, suitable, and reviewable to do so.

Notebook 2 (N2) exists to formalize that line as a gate. The architecture is a suitability boundary that decides, with explicit rules and auditable traces, whether an incoming request can proceed through an AI-assisted workflow, must be redirected to a different channel, or must be refused. This is not a "compliance module" in the legal sense and it is not a substitute for professional review. It is a disciplined control surface that prevents the institution from treating every question as the same kind of question. It distinguishes between information requests, educational explanations, generic market commentary, and requests that imply a personalized recommendation, a solicitation, or a decision that carries heightened duty of care. The goal is to make safe behavior the default by forcing the workflow to classify the request, test it against policy-aligned criteria, and terminate early when necessary.

The enterprise problem is recurrent because incentives push in the wrong direction. Front-line teams want to be responsive. Stakeholders reward speed and punish delay. Managers want to reduce workload by answering quickly rather than collecting context. Even well-intentioned analysts can drift into recommendation language when they are asked for "the best option." A general-purpose language model amplifies this drift because it is trained to be helpful and to fill in gaps. If the institution deploys LLMs without an explicit suitability boundary, the model will often generate text that reads like advice even when the user did not provide the minimum information that would justify advice. That produces two failures at once: the organization gives an answer it cannot defend, and it creates a record that appears to show the organization endorsing an action.

The N2 architecture addresses this by treating suitability as an explicit decision point, not as a paragraph at the end. The workflow begins by normalizing the request: what is being asked, who is asking, and what

outcome the requester expects. It then evaluates the request through a set of hard branching rules that represent institutional constraints. Some rules are about content: explicit requests for guaranteed returns, leverage instructions, or circumventing controls should trigger refusal. Some rules are about context: high-stakes decisions, vulnerable users, or ambiguity about intent should trigger human review. Some rules are about scope: if the request is educational ("explain how mortgages work"), the workflow may proceed in an informational mode. If the request is personalized ("tell me what I should do with my portfolio"), the workflow should redirect to a qualified professional process and stop generating recommendation-like language.

In a firm, this boundary must be defensible in front of a committee. That is why N2 is designed to produce a clear deliverable: a suitability decision record. The record states the classification, the triggered policy reasons, the chosen route, and the permitted response mode. If the system proceeds, it does so under a constrained mode such as general education or generic information with explicit uncertainty and without personalization. If the system redirects, it produces a handoff note that explains what information is required and which human channel should take over. If the system refuses, it provides a standardized refusal message that is policy-aligned and avoids escalating conflict. The institution's objective is consistency: similar requests should be treated similarly, and the reasons should be visible.

This matters because in real operations the most dangerous category is the "almost advice" request. The user does not say "recommend," but the organization's response can still be interpreted as a recommendation. A question like "Is this a good rate?" can be answered as a market fact ("rates in this band are common this week") or as a suitability statement ("this is good for you"). Without a boundary, the model may drift into the second. The N2 gate prevents that drift by forcing the workflow to choose a mode before generating content. In a committee context, mode selection is a control: it limits what the model is allowed to output. It also structures human responsibility: if the mode is informational, the output can be reviewed as education. If the mode is recommendation-adjacent, the workflow terminates into human review.

The architecture is also designed to be transparent about uncertainty. Suitability is rarely binary in the real world. Many requests are ambiguous, and ambiguity is itself a risk. The gate therefore includes a conservative branch: when the system cannot confidently classify the request as safe to handle in informational mode, it routes to HUMAN_REVIEW. This branch is not a weakness; it is the institutional posture that recognizes that a safe system must fail closed. It is better to escalate than to produce advice-like text under uncertainty.

From a board perspective, the value of N2 is that it converts a vague, cultural notion of "be careful" into a measurable process. The institution can track how many requests are routed to refuse, redirect, or proceed; which policy reasons trigger most often; and where front-line teams are repeatedly asking for outputs that exceed scope. Over time, this becomes a governance asset: it reveals where training is needed, where intake forms can be improved, and where the organization should not deploy automation at all. The gate also reduces model risk by reducing exposure: many high-risk requests are terminated before the model is asked to generate a long response.

The failure modes that N2 is explicitly designed to address are operational and common. First is suitability leakage: the system proceeds as if it is in informational mode but outputs language that implies individualized

advice. Second is inconsistent handling: two similar requests receive different routing because classification is implicit rather than explicit. Third is silent policy drift: prompts change, model versions change, and the behavior shifts without a record of why. Fourth is overconfidence under missing context: the model infers time horizon, risk tolerance, constraints, and then recommends. Fifth is stakeholder misinterpretation: the requester treats the output as an approval because the system did not clearly label limits and did not route to human review. N2 mitigates these by making the decision explicit, by logging the decision, by bounding the generation modes, and by terminating early when needed.

The governance posture is deliberately conservative because the board is accountable for outcomes, not intentions. The architecture assumes that if a request can be interpreted as advice, it will be interpreted as advice by someone, somewhere, at some later date. The right control is therefore to prevent the system from creating advice-like artifacts unless the organization has put human review and qualified responsibility in place. N2 does not try to solve the entire compliance universe. It solves a narrower, essential problem: it makes the workflow aware that there are categories of requests that must not be handled by an automated drafting path.

In practical operations, the N2 gate can be deployed as a front door. Any workflow that uses an LLM to assist with financial communication should pass through N2 first. That includes personal finance triage, investment commentary generation, internal memo drafting, and client email assistance. The gate does not need to know all details to be useful. It needs to know enough to separate low-risk informational requests from high-risk personalized requests, and it needs to do so consistently. The key is not perfect classification; the key is conservative classification with explicit escalation.

The decision packet for the board is therefore framed around what to approve, what to forbid, and what to require. Approve a pilot where N2 is used as a mandatory pre-check for any LLM-assisted response in the personal finance domain. Require that the gate's routes are limited to three outcomes: PROCEED_IN_INFORMATIONAL_MODE, REDIRECT_TO_QUALIFIED_CHANNEL, or REFUSE_WITH_STANDARD_LANGUAGE, plus an explicit HUMAN_REVIEW fallback when classification is uncertain. Forbid the system from producing personalized recommendations or action instructions in the proceed path. Require artifact logging as a release criterion so that every decision can be audited. Require metrics in pilot: refusal rate, redirect rate, human review rate, and downstream overrides by reviewers.

A conservative deployment also requires institutional discipline about labeling. When the system proceeds in informational mode, the output must be labeled as general information, not advice, and must avoid individualized language. When the system redirects, the output must explain what information is needed and who is responsible. When the system refuses, the output must be polite, short, and consistent. These are not stylistic choices; they are governance controls that reduce misinterpretation risk. They can be standardized and reviewed once rather than reinvented for every case.

Finally, N2 changes how the organization thinks about AI: it treats "not responding" as a valid system outcome. In many firms, refusal is socially hard; people fear that refusing looks unhelpful. N2 normalizes refusal and redirection as professional behavior when the request is unsuitable. That is a cultural shift, and it is precisely

the shift that governance-first architectures are meant to enable. The board should approve N2 not because it generates better answers, but because it reduces the probability that the firm produces the wrong kind of answer in the wrong context, and because it provides a traceable record of the decision to stop.

To make the scenario even more realistic, consider the internal chain of custody of a message. A stakeholder message is often copied into chat channels, ticketing systems, and email threads. Each hop strips context and increases the chance that someone responds out of scope. In that environment, the suitability boundary acts like a circuit breaker: it forces the responder to declare the category of the response before producing content, and it produces a small decision record that can be attached to the ticket. When a supervisor later asks why a request was escalated, the team can point to the gate decision and the policy reasons, rather than reconstructing intent from memory.

Executives typically ask three questions when they review such a control. First, does it slow us down? The answer is that it changes where time is spent: it reduces time spent drafting risky prose and increases time spent clarifying scope, which is exactly what should happen. Second, will it annoy stakeholders? In practice, consistent redirection is less annoying than inconsistent partial advice, because it sets expectations and creates a clear next step. Third, can we prove it worked? The artifacts and metrics answer that: the institution can show how many times it refused, redirected, or escalated, and it can sample cases to confirm that the gate prevented advice-like language in proceed paths.

The pilot should also include explicit override handling. Humans will sometimes override a redirect and proceed anyway, or override a refusal. That is acceptable only if overrides are logged with a reason and a reviewer identity, because overrides are high-signal governance events. A mature version of N2 would treat overrides as training data for policy refinement, not as silent exceptions. In early pilots, the board should require that any override triggers a brief post-mortem: what misclassified the request, what policy was unclear, and what change will prevent recurrence.

In effect, N2 converts suitability from a subjective feeling into a workflow. That is the essence of governance-first design: define the boundary, enforce the boundary, record the boundary decision, and improve the boundary with feedback.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

The suitability boundary is implemented as a LangGraph workflow whose purpose is to make routing explicit and auditable. The central idea is that an LLM can draft language, but it must not decide, implicitly and invisibly, whether it is allowed to draft. In N2, that permission decision is separated from generation and encoded as deterministic control flow over an explicit state.

The workflow begins with a TypedDict state that acts as the decision ledger. At minimum, the state carries the raw user request, a normalized interpretation of what is being asked, a tentative request category, a risk tier, a list of triggered policy flags, a selected response mode, and a final routing decision. The state also includes termination metadata such as a stop reason and a trace of which nodes were executed. Because the state is

typed and explicit, the system cannot "remember" a classification unless it writes it into a field that will later be exported into final_state.json.

A node is a single responsibility function that reads from state and writes back to state. In this project, nodes are wrapped in an AgentNode abstraction so that every node has a name, a clear input/output contract, and consistent logging. The distinction between node and agent is practical: an agent node may call an LLM for classification or drafting, but it is still bounded by the graph. It cannot create new routes; it can only output fields that downstream gates will evaluate.

Edges connect nodes. Some edges are unconditional, used for initialization and packaging. The important edges are conditional edges, which implement gates. A gate is a deterministic function that inspects the state and chooses the next node. In N2, the primary gate is the suitability gate. It takes as input the request category, the policy flags, and any uncertainty indicators, and it outputs a route such as PROCEED_IN_INFORMATIONAL_MODE, REDIRECT_TO_QUALIFIED_CHANNEL, REFUSE_WITH_STANDARD_LANGUAGE, or HUMAN_REVIEW. The key governance decision is that the route is not selected by free-form model text; it is selected by explicit rules evaluated over structured fields.

Bounded loops appear in N2 as a conservative retry mechanism for classification. Ambiguous requests can trigger a limited number of clarification attempts. The loop counter is stored in state, and the graph enforces a maximum. When the loop cap is reached, the system terminates into HUMAN_REVIEW rather than continuing to ask questions or drifting into advice-like drafting. This bounded loop is a control: it ensures the workflow remains predictable under adversarial or unclear inputs.

The boundary between hard-coded code and model use is explicit. Hard-coded code implements the safety surface: state schema, default values, policy flag definitions, gate logic, route mapping, loop caps, and termination reasons. These are the elements the committee should review and approve as institutional policy encoded in software. The LLM is used only for bounded language tasks: interpreting the user request into a normalized form, suggesting a request category from a controlled taxonomy, optionally drafting a short clarification question, and drafting standardized output language that is consistent with the selected response mode. Even when the LLM drafts language, the workflow constrains it by mode. For example, in informational mode the drafting prompt forbids personalized instructions and requires general education framing, while in refusal mode it requires short, polite, policy-aligned language.

Artifacts are the mechanism that makes this architecture reviewable. run_manifest.json captures run identifiers, configuration hashes, environment fingerprints, and versioning so the decision can be reproduced. graph_spec.json captures the graph topology, node names, and conditional routes so the permitted behavior is explicit. final_state.json captures the actual decision ledger: the input request, the classification fields, the policy flags, the selected route, the stop reason, and the final drafted response text if one was produced. Together, these artifacts allow a risk team to sample cases and answer concrete questions: did the workflow proceed when it should have redirected, did it refuse when it should have escalated, and did the proceed path remain within informational constraints.

Human intervention is treated as a designed endpoint, not a contingency. The HUMAN_REVIEW route is

a first-class termination that packages the decision record and specifies why a human must take over. In a production setting, humans can also intervene by overriding a route, but overrides should be logged as part of the state or as a linked review artifact, because overrides are governance events. The architecture therefore encodes a conservative principle: models can draft, gates decide, humans approve. This separation is what makes suitability a controllable property rather than an emergent behavior.

Mermaid visualization plays a practical governance role in this notebook. By rendering the graph as a diagram, the workflow becomes legible to non-engineers: the committee can see where classification occurs, where the gate branches, where clarification loops can repeat, and where the process terminates. The diagram is not the source of truth, but it is a review aid that matches the stored graph specification. When policy changes, the institution can review the updated diagram and the updated graph_spec.json together to confirm that the routing surface still matches the approved control intent.

Because suitability rules may evolve, the architecture is designed to keep policy parameters in configuration rather than in scattered prompts. The taxonomy of request categories, the mapping from flags to routes, and the maximum retry counts should be versioned and hashed into the run manifest. This makes "policy drift" measurable: if the refusal rate changes, the organization can distinguish changes caused by model behavior from changes caused by policy configuration. A disciplined team can also build small regression suites: synthetic requests that should always refuse, always redirect, or always proceed in informational mode. Running those suites on every change provides a lightweight form of control testing that is aligned with how software and model risk are managed in practice.

This is why the gate is reviewed like code: explicit, testable, and accountable to owners.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

The most important improvement to the N2 architecture is to treat it as a living control, not a one-time build. Suitability boundaries deteriorate when they are not monitored, when language modes drift, or when teams learn informal workarounds. The board should therefore require a governance cadence. Weekly governance reviews keep the boundary calibrated continuously: periodic review of routing rules, periodic sampling of artifacts, and explicit ownership for changes. Without ownership, the boundary will become porous, and the institution will slowly return to the default behavior of answering quickly and hoping for the best.

A second improvement is to formalize the request taxonomy and risk tiers with the same discipline used for financial product taxonomies. The system should not rely on vague labels such as "safe" or "unsafe." It should use an agreed set of categories that map to institutional obligations and channels. For example, informational education can be a permitted category; personalized recommendation can be a restricted category; and ambiguous mixed requests can be a default escalate category. The taxonomy should be documented and versioned, and the mapping from category to response mode should be reviewed by stakeholders across functions, including risk, compliance, and front-line leadership. Where legal interpretation is required, that remains an **Open item for compliance review**, but the operational taxonomy can still be governed internally.

A third improvement is to tighten language controls in proceed mode. In practice, most suitability failures occur not because the route was wrong, but because the language in the proceed path drifted into individualized phrasing. The remedy is to standardize a "safe style guide" for informational responses. It should forbid second-person imperatives, prohibit single-action prescriptions, and require statements that highlight uncertainty and the need for professional review when decisions are high-stakes. This style guide can be implemented as prompt constraints and also as deterministic post-checks that flag prohibited patterns. The board should prefer simple, testable constraints over complex heuristics.

A fourth improvement is to integrate structured intake signals from N1 and other workflows. Suitability cannot be assessed purely from the phrasing of the question; it is also a function of what information is missing. If an intake packet shows that time horizon, liquidity needs, and constraints are unknown, the default should be to escalate. Over time, the institution should connect suitability gating to completeness gating, so that "proceed" requires both a permitted category and minimum information sufficiency. This creates layered controls: N2 decides whether the organization is allowed to respond in an automated way, and N1 decides whether the organization has enough information to respond responsibly even in informational mode.

A fifth improvement is to define and monitor pilot metrics that are meaningful to governance. At minimum, the board should require five metrics. First, redirect rate: the proportion of requests routed to a qualified human channel. Second, refusal rate: the proportion refused under policy flags. Third, human review rate: the proportion escalated due to ambiguity or high-risk context. Fourth, loop exhaustion rate: the frequency with which clarification attempts hit the cap, indicating either poor question design or unclear stakeholder intent. Fifth, override rate: how often humans override a gate decision, which is a leading indicator of miscalibration. These metrics should be tracked by request category and by channel, because a gate that behaves well in one channel may behave poorly in another.

In addition to metrics, the board should require sampling and qualitative review. A small monthly sample of cases should be reconstructed from final_state.json and reviewed by a cross-functional panel. The panel should ask three questions: did the route match the policy intent, did the language stay within the allowed mode, and did the system stop and escalate when uncertainty remained. The artifact bundle makes sampling feasible.

A conservative deployment plan for N2 starts with internal use. Deploy the gate as a mandatory pre-check for staff drafting support, not as a client-facing interface. Require that outputs are attached to internal tickets and that no client communication is sent without human review. After the pilot, expand only if metrics are stable and if sampling shows consistent behavior. The institution should treat expansion decisions as governance decisions rather than as feature rollouts.

The board should also insist on adversarial and stress testing. Suitability gates are vulnerable to prompt injection, social engineering language, and requests framed to bypass controls. The institution should test scenarios where the requester pressures speed, claims expertise, or tries to make the system "just tell me what to do." The expected behavior should be consistent: the system either stays informational with strict language constraints or redirects and refuses. These tests can be automated using synthetic prompts and

should be rerun whenever model versions or prompts change. The goal is not to eliminate every bypass, but to demonstrate that the institution tests the boundary and improves it.

Finally, N2 should be positioned as a governance foundation for the entire agentic program. Every later notebook in this handbook becomes safer if suitability boundaries exist upstream. Credit memo drafting, trading hypothesis workflows, and research synthesis all become less risky when the system can refuse and redirect early. The board should therefore view N2 as a control component that can be reused across workflows, not as a one-off feature. Approving N2 is approving a principle: not every question is permissible, not every output is appropriate, and the system must be designed to stop.

The recommendation is clear. Approve N2 as a mandatory suitability gate for LLM-assisted financial communications in the pilot scope. Require explicit routes, bounded clarification, and artifact export for every run. Forbid personalized recommendations in proceed mode and require a safe informational language standard. Require monitoring metrics and monthly artifact sampling. Require documented ownership for policy changes and regression tests before updates. With these conditions, the institution does not merely adopt an AI tool; it adopts a reviewable control that reduces the probability of advice-like outputs emerging from uncontrolled interaction. That is the board-level outcome that matters: a process that fails safely, records its reasons, and protects the institution's duty of care under uncertainty.

Chapter 3

Credit Memo Drafting with Evidence Gaps

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_3.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

A credit memo is not a document; it is an accountability artifact. In a financial institution, the memo is the object that allows a committee to say, later and under scrutiny, why a decision was made. When credit goes bad, the memo is re-read. When a portfolio is reviewed, the memo is audited. When limits are questioned, the memo becomes the evidence that the process was disciplined. And yet the recurring problem inside firms is not the absence of writing capacity. It is the absence of evidence completeness. Credit memos are drafted under time pressure with partial data: management calls have not happened, audited financials are delayed, covenants are ambiguous, collateral details are incomplete, and market comparables are stale. Analysts are forced to move forward anyway, because deals do not pause for perfect information. The operational risk is therefore not that the memo contains an error. The operational risk is that the memo disguises uncertainty, converts missing information into confident narrative, and prevents a committee from seeing what is actually known.

Notebook 3 (N3) exists to address that exact failure mode: the conversion of evidence gaps into prose. The architecture is a governed credit memo drafting workflow that treats "missing evidence" as a first-class object, not as an inconvenience. Its core deliverable is not just a draft memo; it is a draft memo accompanied by an evidence gap register that explicitly separates facts, assumptions, and open items. The workflow is built to behave like a disciplined internal team: one role drafts the memo using the available facts, another role critiques the memo by checking whether claims are supported, a third role maintains a gap ledger, and a final gate decides whether the memo is suitable to submit to the credit committee, must be revised, or must be escalated for human review due to unresolved high-risk gaps.

The institutional scenario is familiar. An analyst is told to "get the memo out" for a borrower renewal, a new facility, or a waiver request. The deal team has a narrative and a timetable. Risk wants to see covenant headroom and downside scenarios. Legal wants clarity on security and priority. Portfolio management wants comparables and margin of safety. Operations wants documentation status. The analyst has some materials—perhaps an initial information package, some internal history, a set of financial statements, and a few notes from a call—but critical items remain uncertain. Under these constraints, the natural human failure is to write as if certainty exists. The memo reads cleanly because the analyst is skilled, but it is clean because it hides complexity. The committee then approves on the basis of a story rather than a controlled evidence

ledger.

An LLM without governance amplifies this risk. The model will fill in missing items with plausible assumptions, and because the prose is fluent, reviewers may not notice. It may state covenant terms, leverage levels, or market dynamics with unjustified certainty. It may also smooth over contradictory evidence by choosing one interpretation. In the credit domain, this is dangerous because the cost of error is asymmetric. A false positive—approving a borrower because the memo sounded confident—can create real losses. Even a correct decision can become indefensible if the institution cannot show what evidence was available and what was assumed.

N3 is designed to prevent that by forcing the memo to be drafted and evaluated as a controlled process with explicit gates. The key idea is that "drafting" and "verification" must be distinct steps. The system drafts a memo, then critiques it using a structured critique loop that checks the memo against the evidence provided. The critique does not merely say "this is weak." It identifies specific claims that lack support and assigns them to the gap register. The workflow then revises the memo to reflect the true evidence status: supported facts remain, assumptions are labeled as assumptions, and open items are elevated as unresolved. The process repeats under a bounded loop. If the memo reaches a defined quality threshold, it can be packaged as "committee-ready draft." If it fails to reach that threshold within the loop bound, the workflow terminates into escalation: human review is required, and the deliverable is a memo draft plus a prominent gap register and stop reason.

This architecture responds to a real committee need: committees do not want certainty; they want disciplined uncertainty. A well-governed memo does not pretend to know what it cannot know. It tells the committee what the institution is betting on, what could falsify the thesis, and what evidence is missing. That is the difference between an informative memo and a persuasive memo. N3 is built to produce informative memos.

The "evidence gap" lens is therefore the center of the enterprise narrative. In a typical firm, evidence gaps appear in predictable places: updated financial statements, quality of earnings, customer concentration, backlog sustainability, collateral valuations, legal enforceability, covenant definitions, intercreditor terms, and sponsor support. Analysts often have partial evidence in each of these categories. The danger is to combine partial evidence into a definitive claim. N3 blocks that behavior by maintaining a ledger: every claim the memo makes must be traced to a supporting fact, labeled as an assumption, or flagged as an open item requiring further diligence. This is not a philosophical stance; it is a risk control.

The failure modes N3 is designed to address are explicit and operational. First is hallucinated support: the memo contains claims that appear factual but are not supported by provided evidence. Second is assumption laundering: assumptions are written as facts through subtle language choices. Third is narrative dominance: the deal story overwhelms downside analysis because it is easy to write and hard to verify. Fourth is inconsistency: different parts of the memo imply different numbers or terms because the drafting process is not controlled. Fifth is committee misinterpretation: a clean memo can be mistaken for a complete memo. N3 mitigates these by forcing critique, by writing the gap register, by bounding the loop, and by producing an explicit termination reason.

The committee's decision packet in this chapter is again conservative. The recommendation is not to approve "automated credit decisions." The recommendation is to approve a governed drafting workflow that improves memo quality and reviewability while preserving human authority. Approve a pilot where the output is a draft memo plus a mandatory gap register and where a human credit officer signs off before the memo is submitted to committee. Restrict the system's scope to drafting and internal critique, not final approval. Require that the system never fabricate missing evidence and that all gaps are surfaced explicitly. Require that artifacts are produced for audit: run manifest, graph spec, final state, and the memo and gap register as deliverables.

This structure matters because credit committees operate under time pressure but also under legal and reputational scrutiny. When a borrower defaults, the institution must show that it followed a disciplined process. A memo that hides uncertainty is a liability. A memo that exposes uncertainty is a control. N3 operationalizes that principle.

The architecture can also improve organizational learning. Over time, the gap register becomes a dataset: which evidence gaps recur, which desks repeatedly submit incomplete packages, which borrower types create persistent uncertainties, and which claims are most often unsupported. This supports governance at the portfolio level: the institution can adjust diligence checklists, strengthen covenants, or alter underwriting standards based on systematic evidence gaps. That is a board-level benefit: better process discipline and better institutional memory.

Finally, the workflow explicitly includes human intervention points. Humans intervene by providing missing evidence, by approving or rejecting the memo as committee-ready, by overriding the workflow when policy requires it, and by deciding whether to proceed with a decision under uncertainty. N3 does not remove human judgment; it makes it better informed and better documented.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

N3 is implemented as a LangGraph workflow with explicit state and explicit routing. The state is defined as a TypedDict and acts as the case ledger for the memo drafting process. At minimum, it contains the raw borrower package inputs, extracted key facts, the current memo draft, a list of memo claims or sections, a gap register that records unsupported claims, an iteration counter for the critique loop, a quality or readiness indicator, termination reason, and final routing decision. The key design choice is that the gap register is part of state, not an external note. That means the gap register is exported in final_state.json and becomes auditable.

Nodes are single responsibility units wrapped in an AgentNode abstraction for consistent logging. A drafting node uses the provided evidence to produce a structured credit memo draft. A critique node reviews the memo draft and checks whether statements are supported by evidence fields in state. A gap extraction node writes unsupported claims into the gap register and classifies them as assumptions or open items. A revision node updates the memo to reflect the gap register, labeling assumptions and escalating open items. A gating node evaluates whether the memo is "committee-ready draft" under defined criteria: clarity of thesis, explicit downside, consistent numbers, and an acceptable residual open-items set. The graph then either terminates

with a packaged deliverable or loops back into another critique iteration.

The routing logic is designed to be deterministic and reviewable. The loop is bounded by a maximum number of iterations stored in state. Each pass updates the gap register and the memo, and the gate decides whether to proceed or iterate. If the loop reaches its cap, the system terminates into HUMAN_REVIEW with a stop reason such as MAX_ITERATIONS_REACHED. This is the fail-closed behavior that prevents the workflow from endlessly rewriting until it produces a persuasive memo. The objective is not persuasion; it is evidence discipline.

The boundary between hard-coded code and LLM use is explicit. Hard-coded logic governs the control surface: iteration bounds, route mapping, termination reasons, schema enforcement, and artifact export. These are institutional controls implemented as code. The LLM is used for bounded synthesis tasks: drafting the memo in a consistent structure, identifying which statements are unsupported relative to the evidence fields, summarizing gaps in plain language, and rewriting sections to include explicit uncertainty labels. Every model output is written to state, so that it can be reviewed and compared across iterations.

Artifacts are central to governance. run_manifest.json records run metadata, configuration hashes, and environment fingerprints to enable reproducibility. graph_spec.json captures the node topology and conditional routing, making the process transparent. final_state.json captures the end-of-run ledger: the final memo draft, the final gap register, the number of iterations executed, the stop reason, and the routing decision. In addition, the notebook should produce deliverables such as a memo file or a memo text blob and a gap register file. These deliverables are the objects the credit committee would actually read, but the artifacts are what allow audit to reconstruct how the memo was produced.

Human intervention is a designed endpoint. The workflow is intended to hand a draft memo and gap register to a qualified human credit officer. The human can decide to request more diligence, accept assumptions explicitly, tighten covenants, reprice risk, or decline the transaction. The architecture does not attempt to replace these decisions. Instead, it ensures that the human sees uncertainty clearly and that the institution can later show that uncertainty was recognized.

The Mermaid graph rendering is a practical governance tool. It makes visible the drafting–critique–revision loop and the gate that controls termination. For a committee audience, this diagram explains in one view that the workflow is not a single model call; it is a bounded process with explicit stops. The diagram should be consistent with graph_spec.json so that governance reviewers can match the visual representation to the machine-readable specification.

Overall, N3 is a controlled writing system whose purpose is to prevent evidence gaps from becoming invisible. The key architectural idea is to encode the gap register as a state object, enforce critique through bounded loops, and terminate with explicit stop reasons and artifacts that allow post-hoc review.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

The next step for N3 is to treat it as a production-quality drafting control that must be calibrated, monitored, and integrated into the institution's credit governance. The first improvement is to define a formal memo schema and gap register schema that align with the committee's decision process. Committees typically want consistent sections: borrower overview, facility terms, business model, financial summary, risks, mitigants, covenants, collateral, sponsor support, downside scenarios, and recommendation. The drafting node should be constrained to that schema. The gap register should have stable fields: claim identifier, memo section, claim text, evidence support reference (if any), classification (assumption vs open item), materiality (low/medium/high), and required diligence action. A stable schema allows committees to read faster and reviewers to compare across deals.

The second improvement is to tighten evidence linking. Even without full document retrieval, the workflow can require that each quantitative claim cite a specific evidence field in state. This can be implemented as a deterministic check that flags any number not present in the evidence dictionary. The objective is not to build a full citation engine; it is to prevent silent invention of numbers. Where external evidence is required, that becomes an open item for compliance and diligence review.

The third improvement is to calibrate stop criteria. "Committee-ready draft" should not mean "no open items." In real deals, some open items remain until closing. The question is whether the open items are understood, bounded, and priced into the decision. The gate should therefore accept a memo if high-materiality open items are explicitly labeled and accompanied by mitigation or conditionality (e.g., covenant tightening, pricing adjustments, closing conditions). Conversely, the gate should force escalation if a high-materiality open item is unresolved and not mitigated. These criteria should be configured and versioned so that policy changes are auditable.

The fourth improvement is reviewer feedback integration. After the credit officer reviews the memo and gap register, the officer should mark which gaps were correctly identified and which were missed, and which memo sections were misleading. These annotations should be captured in a review artifact and, where possible, fed back into prompt governance and rule refinement. Over time, this will reduce false positives (irrelevant gaps) and false negatives (missed critical gaps). The institution should treat this as part of model risk management: the goal is continuous improvement under governance.

The fifth improvement is monitoring metrics that map directly to process quality. The board should require at least six metrics. Gap density: the number of open items per memo and per section. High-materiality gap rate: the proportion of memos with unresolved high-materiality gaps at termination. Iteration count: how many critique loops were required before termination, indicating drafting stability. Loop exhaustion rate: the percentage of runs that hit MAX_ITERATIONS_REACHED, indicating either poor prompts or insufficient evidence inputs. Consistency error rate: the frequency of mismatched numbers across sections detected by deterministic checks. Human override rate: how often credit officers override the gate decision or materially rewrite the memo, indicating miscalibration.

In addition to metrics, the institution should implement periodic sampling. A monthly sample of memos

should be reconstructed from artifacts and reviewed by a cross-functional panel. The panel should ask whether the gap register genuinely reflects evidence status, whether the memo labels assumptions clearly, and whether any claim appears to be unsupported. This sampling is essential because the biggest risk is not that the memo is wrong; it is that it hides uncertainty.

A conservative deployment plan begins with internal use only. N3 should be used to draft internal memos, not client-facing communications. Every output should be labeled Not verified and should require human sign-off. The workflow should be integrated into existing credit approval processes as a drafting accelerator and a discipline enforcer, not as an approver. The board should also require clear data handling rules: what borrower information may be included, what must be redacted, and what retention policies apply. These are open items for compliance review and must be resolved before production use in sensitive contexts.

Finally, the board should view N3 as part of an institutional posture toward AI: convert narrative into process, and convert uncertainty into explicit registers. Approving N3 is approving a safer way to write. It reduces the probability that the institution will later face scrutiny with a memo that reads confident but cannot show its work. It also creates the foundation for continuous learning: over time, the institution can see which evidence gaps recur and can adjust diligence requirements accordingly.

The recommendation is specific. Approve N3 as a pilot governed drafting workflow for credit memos with the following conditions: bounded critique loops, mandatory gap register output, mandatory artifact export per run, deterministic checks for numeric invention where possible, and mandatory human credit officer sign-off. Forbid the workflow from presenting assumptions as facts. Require monitoring metrics and monthly artifact sampling. Require explicit ownership for schema changes and gate criteria changes. Under these conditions, N3 will not be a flashy AI tool; it will be a process improvement that strengthens committee decision-making, improves defensibility, and reduces a common class of institutional failure: decisions made on the basis of prose that hides missing evidence.

Chapter 4

Trading Hypothesis and Backtest Wrapper (Tool-Augmented)

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_4.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

In an institutional trading environment, the request that triggers most "research automation" is deceptively simple: someone has an idea and wants an answer fast. A portfolio manager says, "I think carry will dominate again if volatility mean-reverts." A strategist says, "This spread looks mispriced; can we test it?" A risk committee member asks, "If we claim this signal works, what happens in the worst regime?" A desk head wants a one-page recommendation before the morning meeting. In every version of the request, the organization is being asked to convert a hypothesis into a claim, and to do it on a timeline that is hostile to discipline. The operational risk is not that the hypothesis is wrong; hypotheses are supposed to be wrong most of the time. The operational risk is that the institution confuses a quick backtest with evidence, confuses a convenient chart with robustness, and confuses a model's confidence with statistical validity.

Notebook 4 (N4) exists to install a governed wrapper around that conversion. The architecture is not "a trading agent." It is a controlled workflow that takes a trading hypothesis, binds it to an explicit specification, runs a backtest through a deterministic wrapper, and produces a decision ticket that can be reviewed by humans. In this system, the model's role is constrained: it helps articulate hypotheses, draft parameter choices, and summarize results, but it does not decide whether a strategy is promotable. The workflow's purpose is to prevent the institution's most common failure in systematic research: treating an unconstrained experiment as an investment process.

The enterprise scenario begins with pressure. A desk has an opportunity, a narrative, or a fear. The desk wants to know whether a signal "works." Often, the definition of "works" is implicit. Is it positive average return? Is it positive Sharpe? Is it stable across regimes? Is it robust to reasonable transaction costs? Is it scalable to the desk's liquidity footprint? Is it uncorrelated to existing exposures? Different stakeholders want different answers, and if those answers are not defined upfront, the backtest becomes a mirror: it returns whatever the researcher is incentivized to see. The main governance problem is therefore specification drift: the hypothesis is refined after results are seen, the sample is filtered after weakness is observed, and the metrics are selected after the curve is attractive. The institution then presents a polished narrative to a committee that cannot tell which parts were predetermined and which parts were retrofitted.

N4 addresses this by enforcing a sequence of actions that resembles a professional research team with a

formal protocol. First, the hypothesis is written down as a contract: instrument universe, signal definition, holding period, rebalancing frequency, constraints, cost model assumptions, and the evaluation metrics that will determine success or failure. Second, the backtest wrapper executes the contract in a deterministic, auditable way. Third, the workflow produces a structured results packet and a decision ticket that can only land in one of a small number of outcomes: APPROVE FOR FURTHER TESTING, REVISE AND RETEST, or REJECT. In practice, this does not "automate investment." It automates honesty: it makes it difficult to run an experiment without declaring what the experiment is.

The distinctive risk N4 is designed to mitigate is that "a backtest exists" becomes a substitute for research rigor. In many organizations, backtests proliferate because they are run and hard to audit. Files live on laptops, scripts differ across analysts, and the definition of costs and slippage is inconsistent. When a strategy later underperforms, no one can reconstruct the exact assumptions that generated the original approval. N4 treats reproducibility as a first-class requirement by generating an artifact bundle on every run. The run manifest captures configuration, environment fingerprints, and a config hash. The graph specification captures routing logic and node topology. The final state captures the hypothesis specification, the executed parameters, the realized metrics, the decision gate outcome, and the explicit stop reason. A committee can therefore ask, "What exactly did we test?" and receive a concrete record rather than a memory.

The backtest wrapper also separates what is deterministic from what is model-driven. The deterministic portion is the simulator: given a price series, a signal series, constraints, and costs, the engine produces returns, drawdowns, turnover, and other diagnostics. The model does not get to change that engine. If the cost model is wrong, it is wrong deterministically, and the correction is a governance decision, not a prompt tweak. The model-driven portion lives around the engine: drafting the hypothesis in structured form, proposing reasonable parameter ranges, and writing plain-language summaries of results and caveats for a committee. This separation matters because it prevents the model from "helpfully" smoothing discrepancies or reinterpreting results. The engine outputs metrics; the model explains them; the gate decides what happens next.

The architecture also includes explicit human intervention points because the institutional role of research is not only to explore but to constrain. Humans intervene at three stages. First, they approve the hypothesis contract before it is executed in any pilot setting, especially when the hypothesis implies material risk or leverage. Second, they review the results ticket, including the open items and caveats, before any next step is authorized. Third, they decide whether a strategy moves to the next research phase, which may include out-of-sample testing, paper trading, execution simulation, or risk committee escalation. N4 is designed to make those human decisions easier and safer by packaging outputs in a consistent, auditable format.

The failure modes that N4 explicitly names are the ones every systematic team has experienced. Overconfidence under limited data is the first: a strong sample period is mistaken for a structural edge. Data leakage is the second: a signal inadvertently uses future information or a survivorship-biased universe. Metric cherry-picking is the third: the strategy is presented using the metric that looks best rather than the metric that is economically relevant. Backtest overfitting is the fourth: parameters are tuned until the curve looks attractive, then presented as if the parameter choice was obvious. Cost blindness is the fifth: turnover and impact are ignored, leading to

paper profits that cannot be executed. Regime fragility is the sixth: performance is dominated by one volatility regime, one liquidity environment, or one macro period. Narrative drift is the seventh: the hypothesis is restated after the fact to match the observed behavior. Finally, stakeholder misinterpretation is always present: executives may treat "the model approved" as an investment endorsement.

N4 mitigates these failure modes by turning them into checks and governance artifacts. The workflow can require that the hypothesis contract include an explicit "what would falsify this" clause: which metric thresholds, which regime failures, or which drawdown constraints would trigger rejection. It can require that the backtest packet include multiple diagnostics: rolling performance, drawdown statistics, turnover, exposure decomposition, and simple stress tests. It can also require that the final decision ticket include open items: for example, "needs out-of-sample validation," "cost model sensitivity required," or "universe definition needs compliance review." This transforms the research conversation from persuasion to process.

From the board's perspective, the core question is what they are being asked to approve. The decision packet is a conservative pilot approval. Approve the architecture as a standardized research wrapper for hypothesis testing, with the explicit restriction that it does not authorize trading. Require that any hypothesis run through this workflow produces the artifact bundle and a decision ticket with one of the allowed outcomes. Require that any "APPROVE" outcome means only "approve for further testing," not "approve for deployment." Restrict the system to synthetic data or approved datasets in the pilot phase, and require human sign-off on the hypothesis contract and on the results ticket. Require metrics for governance: how often strategies are rejected, how often they are revised, and how often approvals are later overturned in deeper testing. This is the only defensible posture for an institution that wants to use AI to accelerate research without weakening controls.

N4 also creates a benefit that is easy to underestimate: it standardizes research language across teams. In a typical firm, analysts write backtests in different styles, making it hard for committees to compare. A governed wrapper produces comparable tickets, comparable metrics, and comparable caveats. That makes oversight more effective. It also makes the organization's learning cumulative. When strategies fail, the artifacts allow the institution to learn which assumptions were wrong and which diagnostics were missing, instead of repeating the same mistakes with new narratives.

Finally, N4 should be understood as a bridge between informal ideation and formal model risk management. Many firms have a gap: ideas are generated in chat, then someone writes a backtest, then someone writes a memo, and the committee sees only the memo. N4 inserts a governed middle layer. It preserves creativity: hypotheses can still be proposed quickly, but it binds creativity to a controlled process that produces reviewable evidence. That is the central promise of this chapter: not that we will find more alpha, but that we will make it harder to confuse curiosity with conviction.

To see the institutional value, consider how these requests are evaluated. A committee rarely asks "is the Sharpe positive?" It asks whether the result is believable and actionable. That means the committee wants to know: was the hypothesis defined before the test, were constraints realistic, were costs modeled conservatively, and were results stable when the sample is perturbed. Without a standardized wrapper, each

analyst answers these questions differently, and committees spend their time deciphering presentation style rather than evaluating evidence. N4 shifts that burden from interpretation to inspection. The decision ticket is deliberately structured to include: the hypothesis contract as executed, the minimum diagnostics required by policy, an explicit list of open items, and a recommended next action. A committee can therefore operate like a committee: it can approve the next step, request a revision, or reject the idea with a recorded rationale.

The workflow also acknowledges that "quick" testing is often the moment where bad incentives enter. If a desk is underperforming, the pressure to find a new signal increases. If a competitor launches a product, the pressure to respond increases. In these conditions, the most common governance failure is not malice; it is drift. Researchers unconsciously choose sample windows that look favorable, exclude periods that look "abnormal," and interpret noise as structure. N4 makes drift harder by recording each choice. Even if the system allows a revise-and-retest path, it records what changed and why. The process therefore creates a cultural safeguard: it treats every revision as an explicit decision rather than a silent tweak.

A board-ready pilot recommendation should also specify what is out of scope. The architecture should not be used to generate live trading instructions, to select position sizes, or to authorize execution. It should not be used with unapproved datasets or proprietary client data without explicit governance. It should not be presented to stakeholders as an "approval engine." Its correct positioning is as a research wrapper that produces internal decision tickets for the next stage in a controlled lifecycle. This out-of-scope list is not bureaucratic; it is what protects the institution from category error: using a research tool as a trading tool.

Over time, the institution can aggregate decision tickets to measure how the research pipeline is behaving: how many ideas are entering, how many are being filtered, and where bottlenecks exist. This aggregate view is governance-relevant because it reveals whether the organization is drifting toward "approval bias" in periods of performance pressure, or toward "rejection bias" in periods of risk aversion. The point is not to optimize for approval rates; it is to keep the pipeline honest and comparable.

The most important point is that the workflow creates a disciplined stopping mechanism. If a hypothesis cannot be specified, the system does not guess; it escalates. If results are sensitive to costs, the system does not hide it; it flags it. If the request is framed in a way that suggests prohibited activity or unsuitable use, the system refuses or routes to human review. These are not features; they are the controls that make acceleration safe. In a governed organization, speed is earned by disciplined stopping, not by optimistic extrapolation.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

The N4 workflow is implemented as a LangGraph state machine whose objective is to make hypothesis testing a controlled, auditable process. The central building block is the explicit state, defined as a TypedDict. Conceptually, the state is a research docket. It carries the hypothesis contract, data references, parameter choices, backtest outputs, diagnostics, decision thresholds, loop counters, termination reason, and final decision. By forcing these elements into named fields, the architecture eliminates hidden memory. If the workflow claims that the hypothesis is "momentum on a 20-day lookback," that claim must be stored as a field. If it claims that the cost model is "X bps per trade plus impact proxy," that must be stored. The

`final_state.json` artifact becomes the ledger that can be reviewed later.

Nodes are single-responsibility units wrapped in an AgentNode abstraction. Each node reads some portion of the state and writes structured updates. A typical node set includes: a hypothesis intake node that normalizes the request; a spec node that constructs the hypothesis contract in schema form; a validation node that checks for missing required fields; a backtest execution node that calls a deterministic simulator; a diagnostics node that computes key metrics; a results summarization node that drafts a human-readable packet; and a decision gate node that assigns an outcome such as APPROVE_FOR_FURTHER_TESTING, REVISE, or REJECT. The point is not the names; the point is that each responsibility is isolated so the committee can understand where judgment is applied.

Edges connect nodes, and conditional edges implement gates. The main gate evaluates whether the hypothesis specification is complete enough to run. If required fields are missing, the workflow routes into a bounded clarification loop or terminates into HUMAN_REVIEW. After backtest execution, a second gate evaluates metrics against predeclared thresholds. If the results violate hard constraints such as unacceptable drawdown, negative performance, or extreme turnover, the workflow can route to REJECT. If results are borderline or sensitive to assumptions, it can route to REVISE with open items. If results meet thresholds, it can route to APPROVE_FOR_FURTHER_TESTING, which is explicitly not a deployment authorization. Each route is a hard branch, recorded in the final state, so that the decision is explainable as policy application, not as model whim.

Bounded loops are used to control iteration. The most common loop in N4 is the "missing spec" loop: the system may ask for clarification on universe, horizon, or constraints, but it can only do so a limited number of times. Another bounded loop may exist for "revise and retest" within a narrow scope, where revisions are constrained to predeclared parameter ranges. The loop bound prevents the classic overfitting failure mode where a researcher keeps tuning until the curve looks good. In a governed architecture, iteration is allowed only within explicit budgets and only with logging of what changed.

The boundary between hard-coded code and LLM use is a primary control. Hard-coded code includes: the simulator and metric computations; the schema for the hypothesis contract; the route mapping and stop reasons; the loop counters and maximum iterations; and the artifact export. These components are deterministic and should be reviewed as institutional policy encoded in software. The LLM is used for bounded language tasks: turning a natural-language hypothesis into a structured contract; proposing candidate parameter ranges; drafting plain-language explanations and caveats; and formatting the decision ticket. The model does not decide the final route. The gate logic decides based on structured metrics and rules. This makes the system auditable and stable under model changes.

Artifacts turn the workflow into an object that can be governed. `run_manifest.json` should capture the run identifier, timestamp, environment fingerprint, package versions, and a config hash. `graph_spec.json` should capture node names, edges, and conditional routing so that permitted behavior is explicit. `final_state.json` should capture the executed hypothesis contract, backtest parameters, outputs, diagnostics, decision thresholds, and the final decision with stop reason. Together, these artifacts support reproducibility and accountability. If

a strategy later fails, the institution can reconstruct what was tested and what was assumed. If a committee asks why a strategy was approved for further testing, the institution can show the metrics and thresholds that triggered the route.

Human intervention is designed as a first-class endpoint. The workflow routes to HUMAN_REVIEW when the request is unsuitable, when data provenance is unclear, when the hypothesis implies prohibited activity, or when loop bounds are reached without completion. Humans also intervene as approvers: a hypothesis contract can require sign-off before execution, and an approval ticket can require sign-off before proceeding to deeper testing. This ensures that institutional responsibility remains human, and that automation accelerates work without displacing accountability.

Mermaid rendering is a governance aid, not a decoration. It makes visible the gates and loops that define safety: where the workflow can ask for clarification, where it can execute the simulator, where it can reject, and where it must terminate. A committee can look at the diagram and understand that "the system" is not a single model call; it is a bounded process with explicit controls and explicit stop reasons. That visual explanation aligns with the stored graph specification, enabling both human understanding and machine review.

In sum, N4 is an agentic architecture only in the disciplined sense: it orchestrates multiple steps under explicit routing. It is not autonomous. It is a governed pipeline that converts a hypothesis into a reproducible test and a reviewable decision ticket. The graph provides structure, the typed state provides a ledger, the nodes provide separation of responsibilities, the gates provide hard policy decisions, the bounded loops prevent uncontrolled tuning, and the artifacts provide auditability. An implementation detail is that the backtest wrapper should expose an explicit I/O contract: inputs are the hypothesis contract and data references; outputs are a results object with returns series, metrics dictionary, and diagnostics bundle. Treating this as a contract allows parallel review and unit testing.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

The board-level objective for N4 is to accelerate hypothesis testing without weakening the institution's standards for evidence and accountability. To achieve that, the next steps focus on tightening controls, improving measurement, and integrating the workflow into the firm's research governance.

The first improvement is to formalize the hypothesis contract schema and treat it as a controlled document type. The contract should include: universe definition, data sources, sampling window, signal computation rules, holding period, rebalancing rule, leverage and exposure constraints, risk controls, cost model, and evaluation thresholds. It should also include a falsification clause: what results would cause rejection. The schema should be versioned and its version embedded in `run_manifest.json` so that changes are traceable. This turns "we tested it" into "we tested this exact contract under this exact policy."

The second improvement is to strengthen data provenance and leakage controls. Many backtests fail not because signals are poor but because the data pipeline is compromised. The workflow should include

deterministic checks for look-ahead bias patterns where possible, enforce a separation between training and evaluation windows, and require explicit declarations of universe construction to reduce survivorship bias. Where these checks rely on institutional data pipelines, their correctness becomes an open item for model risk and data governance review, and the workflow should route to HUMAN_REVIEW if provenance cannot be established.

The third improvement is to expand diagnostics beyond headline performance metrics. Committees are often presented with Sharpe and cumulative return, but what matters operationally is stability and feasibility. The workflow should compute regime-conditioned performance, drawdown statistics, turnover and cost sensitivity, exposure decomposition, and simple stress scenarios. It should also compute fragility indicators such as performance concentration in a small number of dates or assets. The decision gate should use a small set of hard constraints that reflect the desk's risk appetite: for example, maximum drawdown thresholds, turnover caps, and minimum robustness across subperiods. This does not guarantee success, but it prevents obvious fragility from being promoted.

The fourth improvement is to tighten revision policies to prevent overfitting. If the workflow supports a REVISE path, revisions must be bounded by predeclared parameter ranges and limited in count. Each revision should be logged as a structured delta in the state, and the final ticket should show the sequence of revisions. The board should require a rule that any strategy that reaches MAX_REVISIONS routes to REJECT or HUMAN_REVIEW rather than continuing indefinitely. This makes iteration disciplined and prevents "searching for a curve" from becoming a hidden practice.

The fifth improvement is to integrate human review in a way that is efficient and accountable. Human review should not be a vague endpoint. It should be a ticketed action with required fields: reviewer identity, decision, rationale, and required next steps. In pilot, the firm should require human sign-off at two points: approval of the hypothesis contract before execution on sensitive datasets and approval of any "approve for further testing" ticket before moving to paper trading or execution simulation. This ensures that responsibility remains human while allowing the system to accelerate drafting and computation.

The sixth improvement is monitoring and metrics, which must be defined before scale. The board should require at least seven metrics. First, decision distribution: the proportion of hypotheses that are rejected, revised, or approved for further testing. Second, loop exhaustion rate: how often the workflow hits max clarification iterations or max revisions. Third, reversal rate: how often an approved-for-testing strategy fails in the next stage, which indicates false positives in the gate. Fourth, sensitivity rate: how often outcomes change materially under cost model perturbations, indicating feasibility risk. Fifth, regime instability: variance of performance across regimes and subperiods. Sixth, reviewer override rate: how often humans overturn the gate decision. Seventh, reproducibility failures: how often rerunning the same contract produces materially different outputs, which would indicate nondeterminism or data drift.

The seventh improvement is change management. Because the workflow encodes policy in gates and thresholds, changes must be controlled. The institution should define owners for schema, thresholds, and simulator assumptions. Any change should trigger regression runs on a fixed suite of synthetic hypotheses

and known behaviors. The resulting artifacts should be stored and reviewed. This treats the workflow like any other risk-sensitive system: change is allowed, but it is documented, tested, and attributable.

A conservative deployment plan follows from these improvements. In Phase 1, deploy N4 internally as a research assistant for analysts, with labels that outputs are Not verified and require human review. Restrict to approved datasets, and prohibit any use that would directly trigger trading. In Phase 2, integrate with paper trading and execution simulation workflows, but require that the output of N4 is only a ticket that initiates deeper testing, not an authorization. In Phase 3, consider integration into model risk management documentation, where the artifacts become part of the model file. At each phase, require that monitoring dashboards are in place and that sampling reviews occur monthly.

The board should also insist on adversarial and stress testing. Research wrappers are vulnerable to pressure language, selective framing, and requests designed to bypass controls. The institution should test scenarios where the requester pressures speed, claims expertise, or asks the system to "just tell me whether to trade it." The expected behavior should be consistent: the workflow either stays within its research scope with explicit caveats, or it routes to HUMAN_REVIEW or refusal. These tests should be rerun whenever prompts, models, or simulator assumptions change.

The board should understand N4 as a governance instrument, not an alpha machine. The workflow will not guarantee better strategies. What it guarantees is that the institution will have a clearer record of what it tested, what it assumed, and why it promoted or rejected ideas. That clarity is defensible under scrutiny and useful for learning. It reduces the likelihood that the firm will later discover that an approved strategy depended on hidden assumptions, accidental leakage, or unrealistic cost neglect.

The recommended board action is therefore specific. Approve N4 as a standardized hypothesis-to-backtest wrapper for research acceleration under strict guardrails: explicit hypothesis contracts, deterministic simulators, hard decision gates, bounded revision loops, mandatory artifact bundles, and mandatory human sign-off for progression to deeper testing. Forbid interpreting any approval outcome as deployment authorization. Require the monitoring metrics as conditions for scale. Under these conditions, the institution gains speed without sacrificing governance, and it turns the common research risk—confusing a quick backtest with truth—into a controlled process.

Chapter 5

Execution Tactics Under Liquidity Conditions (Regime Machine)

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_5.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

Execution is where elegant strategies go to be humbled. In an institution, a strategy is approved in committee as if it will be implemented frictionlessly: signals generate target positions, positions become orders, orders become fills, and fills become P&L. In practice, the transition from target to trade is mediated by liquidity, volatility, market impact, venue conditions, and timing constraints. The execution desk lives inside the part of the system that is least forgiving and most path-dependent. When liquidity is abundant, errors are expensive but survivable. When liquidity is thin or volatility spikes, the same errors can become existential: trading too aggressively moves the market against you; trading too slowly fails the mandate; pausing too long leaves exposure unmanaged. The recurring enterprise problem is therefore not "how do we trade." It is "how do we adapt execution tactics under changing liquidity regimes while remaining auditable and accountable."

Notebook 5 (N5) exists to formalize that adaptation as a governed workflow. The architecture is a stateful regime machine for execution tactics. It does not attempt to predict prices. It does not attempt to "optimize" in the abstract. It implements a controlled decision loop that chooses between execution tactics—accelerate, slice, pause, reroute—based on observed conditions and explicit risk constraints. The purpose is to prevent a familiar institutional failure: execution decisions made ad hoc under pressure, with no structured rationale and no reproducible record of why a tactic was chosen.

In the real enterprise setting, execution decisions are made in a noisy environment. A portfolio manager may demand rapid rebalancing because the signal has changed. Risk may impose limits because volatility is elevated. The market may show widening spreads and reduced depth. A trader must decide whether to cross the spread, post passively, use darker liquidity, or pause until conditions normalize. These decisions are tactical, but they create strategic outcomes: the cost of execution can dominate the expected alpha, and poor execution can invalidate a strategy that "works" on paper. Yet institutions often treat execution as a human art rather than a governed process. The consequence is that after a poor outcome, the institution cannot answer the question the board inevitably asks: "Why did we trade that way?"

The architecture in N5 is designed to answer that question by design. It treats execution as a controlled loop with explicit state, explicit regimes, explicit constraints, and explicit stop reasons. The workflow begins with an execution objective—an order quantity, a time horizon, and risk parameters—and then runs a bounded

iteration loop. Each iteration observes market condition proxies (liquidity, volatility, spread, urgency), classifies the current regime, selects a tactic consistent with policy, executes a simulated step, updates remaining quantity, and logs the choice and the reason. If conditions deteriorate beyond a defined threshold, the workflow can pause and escalate. If time runs out, it can terminate with a MAX_STEPS_REACHED stop reason. If the order completes, it terminates with COMPLETED. The point is not that the workflow always chooses the best tactic; the point is that it always chooses a tactic through a reviewable process.

The enterprise relevance is direct. Institutions increasingly face scrutiny over execution quality: best execution expectations, internal policies, and client expectations require defensible decisions. Even when a firm is not making a legal claim, it still has an internal duty of care to execute in a way that is consistent with its policies and risk appetite. N5 provides a mechanism to encode those policies. For example, the institution may hard-code that in stressed liquidity regimes, maximum participation rates are reduced, or that in high-volatility regimes, aggressive crossing is limited unless urgency is extreme. These rules are not subjective preferences; they are governance choices that can be reviewed by committees and adjusted over time.

The recurring problem N5 solves is the mismatch between how execution is discussed and how execution happens. In committees, execution is often summarized as "we assumed X bps slippage." In reality, slippage is path-dependent and regime-dependent. A single average number hides whether costs occur due to spread crossing, impact, adverse selection, or timing risk. The architecture forces the institution to confront execution as a sequence of decisions under changing conditions. It produces an execution trace that can be audited: at step 1, regime was NORMAL, tactic was SLICE_PASSIVE, remaining quantity was Y; at step 2, regime shifted to STRESSED, tactic changed to PAUSE; and so on. This trace is the execution desk's equivalent of a credit memo: it is an accountability record.

The distinctive element in N5 is the regime machine. Rather than treating "market conditions" as a continuous blur, the workflow discretizes them into regimes that matter for policy. This can be as simple as NORMAL vs STRESSED vs HALT, or more granular. The key is that the regime classification is recorded and that tactic selection is conditioned on regime. This mirrors how real desks operate: they have playbooks for "normal markets" and "stressed markets." N5 encodes that playbook in a controlled graph.

The failure modes N5 is designed to address are those that cause institutional surprises. Overconfidence in liquidity is the first: executing as if depth exists when it does not. Panic aggressiveness is the second: crossing the spread too heavily under stress, causing impact and adverse selection. Uncontrolled delay is the third: pausing without clear escalation, leading to missed risk reduction objectives. Inconsistent tactics is the fourth: switching tactics without a recorded rationale, making it impossible to learn. Loop runaway is the fifth: continuing to trade or to simulate without a bounded horizon. Stakeholder misinterpretation is the sixth: presenting execution results as "optimized" when they are not. N5 mitigates these by bounding loops, recording regimes and tactics, and producing explicit termination reasons.

From the board's perspective, this architecture is not about replacing traders; it is about making tactical decision-making governable. The decision packet should therefore recommend a pilot where the workflow is used as a decision-support and logging framework, not as an autonomous execution engine. Approve it for

simulation and pre-trade planning: given an order objective and scenario parameters, generate an execution plan and a trace. Require mandatory human sign-off before any live use. Restrict categories such as illiquid instruments or stressed markets to human-only decisions, while still allowing the workflow to produce a trace and recommended actions for review. Require metrics and audit artifacts as release criteria. The point is to embed governance and transparency into execution decisions without pretending that execution can be safely "automated" in the general case.

N5 also creates a disciplined language for cross-functional communication. Risk teams can understand regime labels and constraints. Portfolio managers can understand why execution slowed or paused. Compliance can review whether policies were followed. The workflow thus reduces the tendency for execution to be described as intuition. It makes intuition explicit as policy.

The board should also appreciate that execution is a high-leverage control point. Even modest improvements in execution discipline can dominate the economics of strategy. A research team can spend months improving a signal, only to lose the edge in impact. Conversely, a disciplined execution process can preserve an edge that would otherwise be consumed by costs. N5 is therefore part of a broader governance-first message: strategy is not just signal; it is implementation under constraints.

In practical terms, the deliverable produced by N5 is an execution decision packet: a summary of the order objective, the regime logic, the tactic policy, a simulated trace of decisions, total cost proxies, completion status, and a final stop reason. It is complemented by the artifact bundle that makes it reproducible. The institution can then answer the board's inevitable questions: what did we assume about liquidity, what did we do when liquidity worsened, and how did we know when to stop?

Finally, human intervention is central. Traders intervene by approving or overriding tactics, by changing constraints, and by deciding when to escalate to supervisors. Risk intervenes by setting regime thresholds and participation limits. Committees intervene by approving the policy mapping from regimes to tactics. N5 is designed to make these interventions explicit and traceable. The workflow does not claim authority; it claims structure.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

N5 is implemented as a LangGraph workflow that behaves like a bounded control loop. The explicit TypedDict state is the execution ledger. It contains fields for the execution objective (initial quantity, target completion horizon, urgency), market condition inputs or proxies (spread, liquidity depth, volatility proxy), the current regime label, the chosen tactic, the remaining quantity, the step counter, accumulated cost proxies, termination reason, and final routing decision. By making these fields explicit, the architecture ensures that the execution story is reconstructible from `final_state.json`.

Nodes are single-responsibility functions wrapped as AgentNode components. Typical nodes include: an initialization node that sets the order objective and policy parameters; a market observation node that updates condition proxies; a regime classification node that maps proxies to a regime label; a tactic selection node

that chooses an action consistent with regime and constraints; an execution simulation node that applies the tactic to reduce remaining quantity and update cost proxies; a gate node that checks stop conditions; and a packaging node that produces an execution packet. The model, if used, is used only for bounded drafting and explanation tasks: for example, drafting a human-readable rationale of why a tactic was chosen given the regime and constraints. The model is not used to decide tactics directly in an unconstrained way. The decision is governed by deterministic policy mapping.

Conditional routing implements gates. After each simulated execution step, a stop gate evaluates whether the workflow should continue, pause, or terminate. Conditions include: remaining quantity equals zero (COMPLETED), step counter equals maximum allowed steps (MAX_STEPS_REACHED), regime indicates an unsafe market state (UNSAFE_REGIME), or a risk constraint is violated (CONSTRAINT_BREACH). The gate is deterministic and auditable, ensuring that termination is not a model whim but a policy outcome.

The bounded loop is the heart of the regime machine. The graph cycles through observe → classify → choose tactic → execute step → check stop, with a hard cap on iterations. This boundedness is a governance control: it prevents runaway simulation and prevents the system from endlessly attempting to "finish" an order in an unrealistic environment. When the cap is hit, the system terminates with an explicit stop reason and routes to HUMAN_REVIEW if appropriate.

The boundary between hard-coded code and LLM usage is explicit. Hard-coded code includes: regime threshold rules, policy mapping from regime to allowed tactics, participation limits, step caps, termination conditions, and cost proxy calculation. These are policy and mechanism. The LLM is used, if at all, to draft explanations, to summarize traces, and to produce readable execution packets for committees. Even then, the explanation must be grounded in state fields and the tactic policy; it must not invent market data or imply optimization beyond what the policy provides.

Artifacts are essential. `run_manifest.json` captures the configuration and environment so that the run can be reproduced. `graph_spec.json` captures the workflow topology and routing logic, making it reviewable. `final_state.json` captures the end state: the executed trace summary, the number of steps executed, the sequence of regime labels and tactics, accumulated cost proxies, remaining quantity, and termination reason. In practice, the workflow should also export an execution trace log as a separate deliverable, but even if it does not, the final state should contain enough structured information to reconstruct the path.

Mermaid visualization serves the governance narrative. A committee can see the loop and the stop gates. They can see that the workflow is not a single decision. It is a repeated process with explicit checks and termination endpoints. This makes it easier to approve the architecture as a controlled system: the committee can point to the gates as the places where policy is enforced.

Human intervention is represented in the routing decisions. The workflow can terminate into HUMAN_REVIEW when a stressed regime persists or when constraints are near breach. Humans can override tactics, adjust constraints, or decide to accept execution risk. Those interventions should be logged. In an institutional deployment, any override should be treated as a governance event with an explicit rationale.

In summary, N5 is a stateful regime machine: a graph-structured loop that updates explicit state, selects tactics

under deterministic policy, stops under explicit conditions, and exports artifacts that make the entire process auditable. It is built to turn execution from an intuition-driven sequence into a process-driven sequence while preserving human accountability.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

The board-level value of N5 is improved execution governance: the institution gains a repeatable and reviewable process for adapting tactics under changing liquidity conditions. The next steps should focus on strengthening the policy mapping, improving measurement, and integrating human review and monitoring so that the architecture can be piloted safely.

The first improvement is to formalize the regime taxonomy and thresholds. A regime machine is only as meaningful as its regimes. The institution should define regimes that correspond to real operational playbooks: NORMAL, TIGHT_SPREAD, THIN_DEPTH, HIGH_VOL, STRESSED, and HALT are examples. Thresholds should be based on measurable proxies available to the desk. The mapping from proxies to regimes must be versioned and reviewed. In early pilots, thresholds can be conservative and simple. Over time, they can be calibrated using historical execution data. This calibration is an open item for quantitative validation and should be treated as a model risk activity, not a casual tweak.

The second improvement is to formalize the tactic policy. Each regime should map to a constrained set of allowed tactics, with parameter limits. For example, in STRESSED regime, maximum participation rate should fall, aggressiveness should be limited, and pauses should be allowed. In NORMAL regime, slicing and passive posting may dominate. The policy should include escalation triggers: if a STRESSED regime persists beyond a certain number of steps, the workflow should force HUMAN_REVIEW. These rules should be treated as committee-approved policy, with clear ownership and change control.

The third improvement is to strengthen cost proxy modeling and diagnostics. In the notebook, cost proxies may be simplified. In a production setting, the institution should track components: spread costs, impact costs, timing risk, and adverse selection proxies. The workflow should report these components separately so that committees can understand why costs occur. This also supports learning: if most cost comes from impact, the policy may need to reduce aggressiveness; if most cost comes from timing, urgency rules may need adjustment.

The fourth improvement is to incorporate constraint-aware escalation. Execution is often governed by risk limits: max slippage, max participation, max intraday risk. The workflow should treat constraint approaches as first-class signals. If a constraint is near breach, the workflow should not continue silently; it should escalate. This can be implemented as deterministic checks that route to HUMAN_REVIEW and produce a short escalation note. The board should require that such escalations are logged and sampled in reviews, because they are the high-signal events that indicate the system is operating in stressed conditions.

The fifth improvement is monitoring and metrics. The board should require a dashboard with at least six metrics: completion rate within the target horizon, average and tail cost proxies, regime distribution over

time, tactic distribution by regime, pause frequency and duration, and escalation frequency. In addition, loop exhaustion rate (MAX_STEPS_REACHED) is critical: if the system frequently hits the cap, it indicates that the policy and constraints may be incompatible with the execution objective. That is not a failure of the workflow; it is a discovery about feasibility. The institution should treat that discovery as actionable: adjust objectives, adjust tactics, or accept that the trade cannot be executed safely under the given conditions.

The sixth improvement is human override governance. Traders will override automated recommendations, and they should. But overrides should be structured. Every override should require a short rationale and should be logged in a review artifact linked to the run. Overrides should be analyzed monthly: are overrides occurring in a particular regime, indicating that the policy is too conservative or too aggressive? Are overrides correlated with adverse outcomes? This is how the institution improves the policy mapping without guessing.

A conservative deployment plan begins with simulation and pre-trade planning. Use N5 to generate execution plans and traces for hypothetical or historical scenarios. Require human review of the plan. Compare the plan's predicted costs and regime decisions to historical execution outcomes where possible. Only after the workflow demonstrates stable behavior and useful traces should the institution consider live decision support. Even then, N5 should remain advisory: it proposes tactics and logs rationales, but the trader executes and remains responsible.

The board should also require stress testing of the regime machine. Simulate volatility spikes, liquidity collapses, and sudden spread widening. The workflow should respond by shifting tactics conservatively, pausing when necessary, and escalating rather than forcing completion at any cost. This is the governance posture: in stress, do less, log more, escalate sooner. The board should explicitly approve this posture because it may conflict with short-term performance pressures. A governed institution prefers controlled under-execution to uncontrolled over-execution when conditions are unsafe.

Finally, the board should see N5 as an enabling layer for later architectures. Research workflows that ignore execution are incomplete. Portfolio rebalancing decisions that ignore liquidity are irresponsible. A regime-aware execution process provides the connective tissue between strategy and implementation. By approving N5, the institution is approving the idea that execution is a governed process with artifacts, not an afterthought with anecdotes.

The recommendation is therefore clear. Approve N5 as a pilot execution decision-support and logging architecture under strict guardrails: deterministic regime classification, committee-approved policy mapping from regimes to tactics, bounded loops with explicit stop reasons, mandatory artifact export, and mandatory human sign-off for any use beyond simulation. Forbid representing the workflow as an autonomous execution engine. Require monitoring metrics and monthly sampling of stressed-regime cases. Under these conditions, N5 will improve institutional resilience by ensuring that when markets change, the institution's execution behavior changes in a controlled, reviewable way, and when the institution cannot execute safely, it stops and escalates rather than improvising.

Paste the generated Chapter 5 text here.

Chapter 6

Portfolio Rebalancing: Risk and Cost Perspectives

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_6.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

In institutional portfolio management, the most dangerous moment is not when markets crash. It is when the portfolio is quietly drifting. Drift is not dramatic, which is why it escapes attention. Exposures accumulate because winners grow, losers shrink, correlations shift, and constraints that were true at inception become false in slow motion. Meanwhile, stakeholders demand action: "Rebalance back to target." "Reduce risk." "Harvest gains." "Control costs." These demands are all reasonable individually, and impossible collectively if they are not explicitly reconciled. The recurring enterprise problem is that rebalancing is not a single objective. It is a negotiation between competing perspectives that speak different languages: risk wants exposure control and drawdown protection, trading wants feasibility under liquidity and costs, portfolio construction wants alignment to strategic weights, and governance wants a defensible process.

Notebook 6 (N6) exists to formalize that negotiation as a governed, multi-perspective workflow. The architecture is a parallel committee: multiple specialist "committee members" evaluate the same proposed rebalance from different angles, produce structured critiques, and then an aggregator produces a consolidated recommendation and a decision packet for human approval. The goal is not to automate the rebalance. The goal is to prevent a familiar institutional failure: a rebalance that is justified by one perspective while silently violating another, leading to either hidden risk or hidden cost.

The board and risk committee will recognize the pattern. A portfolio manager proposes a rebalance that restores target allocations and reduces tracking error. The execution desk warns that trading costs will be high, liquidity is thin, and impact will dominate. Risk notes that the portfolio's tail exposure is already elevated and that the proposed trades increase exposure to a stressed factor. Finance notes the constraint that realized gains create tax or accounting implications. Operations notes settlement and operational constraints. In a real firm, these perspectives often collide late, after a plan has already been socially accepted. The result is either an internal conflict that wastes time or a compromise that is not explicitly documented. Both outcomes are governance problems: the institution cannot later show why the final trades were chosen.

N6 makes that collision explicit and early by encoding it as an agentic architecture. The workflow begins with an initial rebalance proposal based on portfolio targets and current holdings. It then routes that proposal in parallel to multiple perspective nodes. Each node behaves like a committee member with a narrow mandate.

A risk perspective node evaluates exposures, concentration, leverage, and drawdown proxies under plausible stress. A cost perspective node evaluates turnover, liquidity capacity, and impact proxies. A portfolio construction perspective node evaluates closeness to target, constraint satisfaction, and alignment with investment objectives. A governance perspective node checks whether the proposal is sufficiently documented, whether assumptions are explicit, and whether the required artifacts are present. Each perspective node returns a structured critique, including: what it likes, what it rejects, what open items remain, and what modifications are recommended.

The aggregation step is the decisive governance innovation. Instead of letting the loudest perspective win informally, the workflow collects structured outputs and consolidates them into a single decision packet. The packet includes dissent and conflicts: if risk wants lower exposure but cost wants fewer trades, the packet makes that trade-off explicit rather than burying it in meeting dynamics. The aggregator then produces one of a small number of routes: APPROVE_AS_PROPOSED, REVISE_AND_REBALANCE, ESCALATE_TO_HUMAN_COMMITTEE, or REJECT. In a pilot posture, approvals should be interpreted as "approve for human sign-off," not as autonomous authorization.

The enterprise rationale is that rebalancing failures are usually failures of process, not failures of mathematics. Many firms can compute optimal trades under a cost model. What they cannot reliably do is align on the objectives, constraints, and risk appetite in a way that is consistent across time and across personnel. People leave, markets change, and undocumented practices become folklore. N6 replaces folklore with artifacts. The system produces a run manifest, a graph specification, and a final state that records the inputs, the competing critiques, the chosen aggregation rule, and the termination reason. This makes the rebalance decision reproducible and auditable.

A second recurring enterprise issue is that costs and risks are not commensurate. Risk is often discussed in exposure units and drawdown probabilities, while costs are discussed in basis points and dollars. Without a structured process, teams talk past each other. The parallel committee architecture does not magically solve that mismatch, but it makes it visible. Each perspective node uses its own metrics but must output its critique in a standardized schema. The aggregator then forces an explicit negotiation: either the rebalance is acceptable under both perspectives, or it is not. If it is not, the system cannot hide that; it must either revise or escalate.

The distinctive value to a board is that N6 operationalizes "second line of defense" thinking inside the workflow itself. In many organizations, risk and cost review occur after a portfolio manager has already committed psychologically to a plan. N6 shifts review earlier and makes it systematic. It also reduces key-person risk. A portfolio manager's judgment is still required, but the workflow ensures that risk and cost critiques are consistently applied even if the personnel changes.

The failure modes N6 is designed to address are concrete. First is single-perspective dominance: a rebalance optimized for target weights ignores liquidity and creates untradeable plans. Second is cost blindness: turnover is treated as an afterthought until it overwhelms expected benefit. Third is risk blindness: rebalancing to targets increases tail exposure under a regime shift. Fourth is false compromise: the team "splits the

difference" without documenting what risk was accepted or what cost was tolerated. Fifth is misinterpretation: stakeholders believe "the system approved" when the system merely consolidated perspectives. Sixth is policy drift: the objective function changes over time without a record, producing inconsistent outcomes. N6 mitigates these by making perspectives explicit, by structuring dissent, by bounding iteration, and by logging artifacts.

The governance emphasis is therefore central. Bounded loops matter because rebalancing can easily become an infinite negotiation: adjust weights, re-evaluate costs, adjust again. N6 permits iteration but bounds it. If a proposal cannot be reconciled within a fixed number of revise cycles, the workflow terminates with MAX_REVISIONS_REACHED and routes to human committee review. This prevents the system from quietly "optimizing" until it finds a narrative. It also creates a measurable indicator: if many cases hit the bound, the firm's constraints may be inconsistent with its targets, or liquidity conditions may make rebalancing infeasible.

The deliverable, again, is not a trade blotter that executes automatically. The deliverable is a decision packet: proposed trades, expected costs, expected risk changes, dissenting critiques, open items, and a recommended route. It is the kind of packet that an investment committee can review quickly because it is structured. This aligns with how boards operate: they want to know what is being proposed, what could go wrong, what was considered, and what controls were applied.

The decision packet to the board should recommend a conservative pilot. Approve N6 for internal decision support in portfolio rebalancing, with mandatory human sign-off. Restrict high-stakes portfolios or illiquid assets to human-only execution decisions, while still using the workflow to structure critiques and produce artifacts. Require that the workflow always exports audit artifacts and that it always includes "facts vs assumptions vs open items." Require monitoring metrics such as turnover, cost sensitivity, escalation rates, and override rates. Require that any "approve" route is interpreted as "approve for human sign-off," not as autonomous authorization.

This architecture also provides strategic benefits beyond individual rebalances. By accumulating critique packets, the institution can learn where its governance friction is concentrated. If cost critiques consistently veto rebalances, the strategic allocation may be misaligned with liquidity reality. If risk critiques consistently object, the risk appetite may be inconsistent with target exposures. N6 thus turns day-to-day rebalance decisions into data for strategic governance. That is a board-level benefit: evidence-based policy refinement.

Human intervention is embedded by design. Humans intervene as policy owners who define constraints, as reviewers who sign off on decision packets, and as arbiters when perspectives conflict. The workflow does not remove these responsibilities. It makes them explicit. It also makes it harder to pretend that a conflict does not exist. If risk and cost disagree, the packet shows the disagreement. That is precisely what the board should want: decisions made with eyes open, not with hidden trade-offs.

In short, N6 exists because rebalancing is not a calculation. It is an institutional conversation. The parallel committee architecture encodes that conversation into a process that can be reviewed, bounded, and audited.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

N6 is implemented as a LangGraph workflow whose defining feature is parallelism. The TypedDict state acts as the rebalance case file. It includes portfolio inputs (current holdings, target weights, constraints), a proposed trade list, market condition proxies (liquidity and cost estimates), risk summary fields (exposures, concentration proxies), the outputs from each committee perspective node, an aggregation record, loop counters for revise cycles, termination reason, and final routing decision. The state is designed so that each committee member writes to its own namespace, preventing accidental overwrites and making dissent traceable.

Nodes are single-responsibility components wrapped in an AgentNode abstraction. The workflow typically begins with a proposal node that generates an initial rebalance plan deterministically or via constrained heuristics. The plan then fans out to multiple committee nodes. A risk node computes exposure changes and flags constraint violations or tail concerns. A cost node computes turnover, impact proxies, and feasibility warnings. A portfolio construction node evaluates target alignment and constraint satisfaction. A governance node checks completeness of documentation, presence of open items, and compliance routing triggers. Each node writes a structured critique to the state: a verdict, key findings, recommended changes, and open items.

The aggregator node is where the architecture becomes a governed committee rather than a collection of opinions. The aggregator reads the critiques and applies a hard-coded decision policy. For example, if any node issues a HARD_BLOCK verdict, the aggregate route cannot be APPROVE; it must be REVISE or ESCALATE. If critiques conflict but none hard-block, the aggregator may propose a compromise modification or route to HUMAN_COMMITTEE. The aggregation policy is deterministic and should be reviewed as institutional governance. The LLM may be used to draft natural-language explanations of the aggregated decision and to summarize dissent, but it does not decide the route. The route is chosen by explicit rules over structured verdicts.

Conditional routing implements revise loops. If the aggregator routes to REVISE_AND_REBALANCE, the workflow enters a bounded revision loop. A revision node modifies the proposed trades according to the highest-priority critiques, then the parallel committee evaluates again. The loop counter is stored in state, and the graph enforces a maximum number of revisions. If the cap is hit, the workflow terminates with MAX_REVISIONS_REACHED and routes to HUMAN_COMMITTEE. This boundedness is a governance control: it prevents endless negotiation and forces escalation when trade-offs cannot be reconciled algorithmically.

Hard-coded constraints are central. They include: max revisions, constraint thresholds, veto rules for committee verdicts, and termination reasons. These define what the system is allowed to do and are the parts the committee should approve. The LLM is used only for bounded tasks: producing readable committee summaries, drafting open-item explanations, and packaging the decision packet. The “intelligence” of N6 is therefore institutional rather than speculative: it is the explicit encoding of how committees actually behave when governed.

Artifacts are mandatory for auditability. `run_manifest.json` captures reproducibility metadata including

configuration hashes and environment fingerprints. `graph_spec.json` captures the parallel fan-out and the aggregation and revision loop topology, making the routing reviewable. `final_state.json` captures the final rebalance proposal, the committee critiques, the aggregation decision, the number of revision cycles executed, the open items, and the termination reason. These artifacts allow post-hoc review: the institution can reconstruct why a particular rebalance was approved, revised, or escalated, and it can test whether policy was applied consistently.

Mermaid rendering helps the board understand that this is not a single model call. The diagram shows the fan-out to committee nodes, the aggregation node, and the bounded revision loop. It is a visual representation of governance: a structured debate that terminates under explicit rules.

Human intervention is represented as the final gate. Even when the workflow routes to APPROVE_AS_PROPOSED, the correct interpretation is "approve for human sign-off" in pilot. Humans review the packet, validate assumptions, and authorize execution. Humans also own the aggregation policy and the veto thresholds. When those policies change, the run manifest and graph specification provide a record of which policy version governed each decision.

In summary, N6 is a parallel committee architecture: explicit state as case file, multiple specialized nodes producing structured critiques, deterministic aggregation rules that encode governance, bounded revision loops, and artifacts that make the process auditable and reproducible.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

The board should treat N6 as a governance instrument for portfolio rebalancing, and the next steps should focus on strengthening policy clarity, improving measurement, and integrating the workflow into the institution's change management.

The first improvement is to formalize the critique schema and verdict taxonomy. Committee outputs should not be free-form text. They should have stable fields: verdict (APPROVE, SOFT_BLOCK, HARD_BLOCK), key risks, key cost concerns, recommended modifications, and open items. A stable taxonomy allows aggregation to be deterministic and testable. It also allows monitoring: the institution can track how often risk hard-blocks versus cost hard-blocks, and where conflicts arise.

The second improvement is to define explicit conflict resolution policies. When risk and cost disagree, the institution must decide what principle governs. For example, risk hard-blocks may always override cost concerns, or cost hard-blocks may override when liquidity makes execution infeasible. These are governance decisions. The aggregator should implement them explicitly and the board should approve them. Where the institution wants discretion, it should route to HUMAN_COMMITTEE rather than embedding discretion in model language.

The third improvement is to calibrate cost and risk proxies. Early notebooks often use simplified proxies, which is acceptable in pilot. For production, the institution must validate that proxies correlate with realized outcomes. This requires historical execution data and risk analytics. Calibration and validation should be

treated as model risk activities with clear ownership. If proxies are uncertain, the workflow should label them as assumptions and increase escalation frequency. This maintains a conservative posture.

The fourth improvement is to integrate scenario and stress testing into the committee evaluation. Rebalances should be evaluated not only under current conditions but under plausible stressed conditions: widened spreads, reduced depth, correlation shocks, or volatility spikes. The risk committee node can simulate simple stress deltas and report whether the rebalance increases fragility. This strengthens governance because it aligns the rebalance decision with the institution's resilience objectives rather than only with baseline optimization.

The fifth improvement is monitoring and metrics. The board should require a dashboard with at least eight metrics. Turnover and predicted cost proxies for each rebalance, and their distribution over time. Risk change metrics such as concentration, factor exposure deltas, and tail proxy deltas. Escalation frequency to HUMAN_COMMITTEE. Revision cycle counts and MAX_REVISIONS_REACHED rate. Dissent rate: how often committee nodes disagree. Override rate: how often humans override the aggregated route. Post-trade realized cost versus predicted cost, where available. Post-rebalance drift rate: how quickly the portfolio drifts back, indicating instability. These metrics map directly to governance: they reveal whether rebalancing is feasible, disciplined, and consistent.

The sixth improvement is to formalize human sign-off procedures. Each decision packet should have a required sign-off field and a reviewer rationale. This should be captured as a review artifact linked to the run. In regulated contexts, this linkage is essential. Even outside regulation, it supports accountability. The board should require that any use of the workflow beyond simulation includes sign-off logging.

A conservative deployment plan should be phased. Phase 1: internal simulation and planning, where the workflow generates decision packets and committees review them in parallel with existing processes. Phase 2: limited live decision support for liquid portfolios with strict guardrails and mandatory human sign-off. Phase 3: broader rollout only if metrics show stable behavior and if sampling confirms that dissent is surfaced and handled appropriately. At each phase, require artifact export, change control, and monthly sampling audits.

The board should also require red-team testing. The most common "governance bypass" in rebalancing is pressure to reduce risk quickly during volatility spikes. This is precisely when costs are highest and liquidity is lowest. The workflow should be tested under such scenarios to confirm that cost concerns are not ignored and that the system escalates rather than silently producing an infeasible trade list. Another red-team scenario is "optimization theater": a rebalance that looks mathematically neat but concentrates risk. The risk committee node should flag this and the aggregator should not approve without escalation.

Finally, the board should view N6 as a reusable pattern. The parallel committee architecture can be applied to many institutional decisions: credit approvals, diligence Q&A, research synthesis. Approving N6 is approving a governance design principle: when objectives conflict, force structured dissent and explicit aggregation under policy.

The recommendation is specific. Approve N6 as a pilot rebalancing decision-support workflow that produces a committee-grade decision packet with structured critiques, deterministic aggregation, bounded revision loops, and mandatory artifacts. Require mandatory human sign-off, restrict high-stakes and illiquid assets to

human-only execution decisions, and require monitoring metrics and monthly artifact sampling as release criteria. Forbid interpreting “approve” as autonomous execution authorization. Under these conditions, N6 will strengthen the institution’s ability to rebalance portfolios in a way that is disciplined, transparent, and defensible, converting an often informal negotiation into an auditable process that the board can support with confidence.

Chapter 7

Investment Banking Pitchbook Sections: Comps, Rationale, Risks

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_7.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

In investment banking, the pitchbook is less a document than a coordination device. It is the object that forces a team to align on a story, align on numbers, and align on what they are willing to defend in front of a client and, later, in front of internal review. The problem is that the pitchbook is always produced under deadlines that are incompatible with careful synthesis. The recurring situation is familiar: a managing director wants a draft by morning, associates are assembling materials, analysts are pulling comps and precedent transactions, and everyone is making small interpretive decisions that can materially change the narrative. In that environment, the operational risk is not that the team is lazy. The risk is that the document becomes internally inconsistent, evidence is scattered across slides without traceability, and the book overstates certainty because persuasion is culturally rewarded.

Notebook 7 (N7) exists to impose a governed structure on that production process. The architecture is a hub-and-spoke constellation that drafts distinct pitchbook sections through specialized nodes, each with a narrow mandate, and then consolidates them through a hub that enforces consistency, surfaces evidence gaps, and packages a review-ready book outline for senior leadership. The goal is not to replace bankers or to generate a full pitchbook automatically. The goal is to reduce the most common institutional failure in pitch work: a polished story that cannot show its work, and a set of sections that were written by different people under pressure with incompatible assumptions.

The enterprise setting is defined by three constraints: speed, narrative coherence, and reputational liability. Speed is demanded by the client cycle. Narrative coherence is demanded by the client's attention and the MD's credibility. Liability arises because even "marketing" materials can be re-read later in disputes, in fairness opinion contexts, or in internal governance reviews. A pitchbook that makes claims about market position, valuation ranges, synergy potential, or risk mitigation must be defensible within the firm's internal standards. Yet the process of producing these claims is often informal. Analysts and associates make quick approximations, reuse prior decks, and stitch together numbers from multiple sources. The result is not necessarily wrong, but it is often not reviewable.

LLMs amplify both the opportunity and the risk. They can draft clean prose, summarize industries, and propose rationales quickly. They can also hallucinate facts, invent comparables, smooth over missing evidence,

and create an illusion of completeness. In a pitch context, the danger is not only factual error; it is the production of confident language where evidence is thin. A senior leader reviewing a deck may not have time to verify every line. If the system produces persuasive sections without surfacing open items, the firm can inadvertently commit to a narrative it cannot defend.

N7 addresses this with a workflow design that mirrors how strong banking teams operate when disciplined. A hub node holds the central case file: the client mandate, the transaction context, the target company profile, and the assumptions that the team is allowed to use. Spoke nodes then generate specific sections: a comps and valuation spoke, a transaction rationale spoke, a risks and mitigants spoke, and potentially a "buyers landscape" or "capital markets conditions" spoke. Each spoke node is responsible for producing a structured section draft and a structured evidence register: what facts it relied on, what assumptions it introduced, and what open items remain. The hub then consolidates these sections, checks for internal consistency, identifies conflicts in assumptions, and produces a pitch packet for MD review.

The distinctive institutional value is the explicit exposure of evidence gaps. In typical pitch work, evidence gaps are handled socially: someone says "we'll confirm later," and later becomes never. N7 turns "confirm later" into open items that must be recorded and either resolved or escalated. This is crucial because many pitch claims are contingent. A valuation range depends on comps selection and normalization. A synergy argument depends on operational feasibility and integration cost assumptions. A risk mitigation claim depends on structuring options and market conditions. Without explicit open items, the deck reads like a certainty machine. With explicit open items, the deck reads like professional work: clear, persuasive, but honest about what is not yet verified.

The failure modes N7 is designed to address are the ones that degrade banking output quality. First is inconsistent assumptions: different sections use different revenue numbers, different growth rates, or different peer sets. Second is false precision: numbers are presented with implied accuracy despite being rough. Third is hallucinated evidence: a model-generated peer or transaction that was not actually provided. Fourth is narrative drift: the rationale section tells a story that the risks section does not acknowledge. Fifth is missing critical caveats: risks are softened or omitted because the deck is a pitch. Sixth is uncontrolled iteration: teams keep rewriting until the deck sounds good, without tracking what changed. Seventh is stakeholder misinterpretation: leadership believes "the system generated this" implies correctness. N7 mitigates these by isolating section responsibilities, requiring evidence registers, and enforcing hub-level consistency checks with explicit termination reasons.

For executive leadership, the key question is what the workflow produces as a deliverable. The deliverable is not a final pitchbook. It is a review-ready pitchbook packet: section drafts, a consolidated assumption ledger, an open-items list, and a recommended next action. That next action may be "ready for MD review," "revise due to inconsistency," or "escalate due to missing critical inputs." In a governed organization, the workflow should default to escalation rather than improvisation when critical inputs are missing. This posture is especially important in banking, where reputational damage often comes from small errors amplified by client scrutiny.

The decision packet to the board should therefore recommend a conservative pilot. Approve the workflow for internal drafting support only, with mandatory human review at the MD or senior associate level. Restrict sensitive claims (valuation ranges, synergy estimates, forward-looking projections) to human-only approval and require that the workflow labels all such content as Not verified unless supported by provided evidence. Require that every run exports the audit artifacts and that every section includes facts vs assumptions vs open items. Require monitoring metrics such as inconsistency rate, open-item density, revision cycle counts, and override rates. Forbid the system from generating "new facts" such as invented comparables or transactions. If the workflow is used with external data sources, that becomes an open item for data governance and compliance review.

N7 also addresses a softer but important organizational issue: standardization across teams. Banking teams often operate with different templates and habits. A hub-and-spoke drafting architecture produces comparable outputs across deals: the same section structure, the same evidence registers, and the same assumption ledger. This makes senior review faster and more reliable. It also improves training: junior staff learn what "good" looks like because the workflow imposes structure.

In the real pitch cycle, the most valuable outcome is not speed alone; it is speed with reduced variance. Leadership wants fewer surprises. N7 reduces variance by enforcing that each spoke node produces not only narrative but also a list of what must be verified. That list changes the conversation. Instead of arguing about whether the deck is "ready," the team can ask: which open items are critical, who owns them, and what is the deadline to resolve them. This is a governance conversation disguised as drafting.

The workflow also creates a record of process. If a pitch fails, or if a client later challenges a claim, the firm can reconstruct how the deck was produced: what assumptions were used, which sections were generated, which inconsistencies were flagged, and what human approvals occurred. This is the board-level relevance: defensibility and control.

Finally, human intervention is integral. Human reviewers decide which comps are appropriate, what valuation messages are acceptable, what risks must be emphasized, and how to balance persuasion with integrity. N7 does not remove those decisions. It makes them explicit by producing structured drafts and structured gap lists that humans can approve or correct. The hub-and-spoke architecture is therefore an institutional mirror: it shows the firm its own process, makes it visible, and makes it governable.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

N7 is implemented as a LangGraph workflow designed as a hub-and-spoke constellation. The TypedDict state is the pitch case file. It contains: the deal context and mandate, the client and target descriptors, any provided financials and key metrics, the approved assumption set, a central open-items list, and per-section outputs from each spoke. Each spoke writes to a dedicated namespace in state, such as `comps_section`, `rationale_section`, and `risks_section`, each with subfields for draft text, facts used, assumptions, and open items. This separation prevents accidental mixing and makes section-level provenance visible.

Nodes are wrapped as AgentNode components with single responsibilities. The hub initialization node loads and normalizes the deal context and constructs the initial assumption ledger. A routing node triggers parallel or sequential execution of spokes. The comps spoke drafts a valuation and comps narrative and produces a comps selection rationale and any missing data items. The rationale spoke drafts strategic logic and deal thesis statements, grounded in provided context. The risks spoke drafts a risk register with mitigants, explicitly distinguishing verified risks from speculative ones. Each spoke node may use an LLM for drafting and synthesis, but it is constrained to write structured outputs into state. It must not invent facts; if evidence is missing, it must write an open item.

A hub consolidation node reads the spoke outputs and enforces consistency. This is where deterministic checks live: do all sections use the same key financial figures, do valuation ranges appear without support, do risk statements contradict rationale claims, are open items duplicated or missing. The hub can also perform a lightweight claim audit: any numeric claim not present in the evidence fields is flagged. Based on these checks, a gate node decides the route: READY_FOR_MD_REVIEW, REVISE_FOR_CONSISTENCY, or ESCALATE_FOR_MISSING_CRITICAL_INPUTS.

Bounded loops apply to revision cycles. If the hub gate returns REVISE, the workflow can enter a limited revise loop where either the hub requests targeted adjustments from specific spokes or a revision node edits the consolidated packet to resolve conflicts. The loop counter is stored in state, and a maximum revision count is enforced. If the cap is reached, the workflow terminates with MAX_REVISIONS_REACHED and routes to human review with explicit stop reason. This boundedness is a control: it prevents endless rewriting and makes escalation explicit.

The hard-coded versus LLM boundary is explicit. Hard-coded code governs: state schema, allowed section templates, consistency checks, revision caps, routing decisions, termination reasons, and artifact export. The LLM is used for drafting within constrained templates and for summarizing open items. The LLM does not choose the route. The gate chooses the route based on structured fields and deterministic checks. This ensures that "ready" is a policy outcome, not a persuasive paragraph.

Artifacts provide auditability. `run_manifest.json` records run id, configuration hash, and environment fingerprint. `graph_spec.json` records node topology, including the hub-and-spoke structure and conditional revision paths. `final_state.json` records the final consolidated packet, each spoke's evidence register, the open-items list, revision counts, and termination reason. Together they provide a defensible record of how the pitch packet was produced.

Mermaid rendering is important for executive understanding. It shows the hub and spokes and the consolidation gate. Executives can see that the process is not a single generation step; it is a coordinated workflow with checks and explicit stops. This aligns with governance expectations: the system is constrained, reviewable, and designed to surface gaps rather than hide them.

Human intervention is represented in the routing decision. READY_FOR_MD_REVIEW does not mean "approved"; it means "packaged for senior review." ESCALATE means the system found missing critical inputs or unresolved conflicts. In pilot, all routes should require human sign-off before any external use. This

preserves accountability and prevents misinterpretation that "the system approved the pitch."

In summary, N7 is a hub-and-spoke drafting architecture: explicit state as case file, specialized section nodes producing drafts plus evidence registers, a hub that enforces consistency and surfaces open items, bounded revision loops, and artifacts that make the process auditable and reproducible.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

The board should evaluate N7 as a governance-first drafting accelerator for investment banking materials. The next steps should focus on strengthening section schemas, tightening fact controls, and integrating the workflow into banking review practice.

The first improvement is to standardize section templates and enforce them through schema. Each spoke should output in a structured format: key messages, supporting facts, assumptions, open items, and a draft narrative. This structure should be consistent across deals and versioned. In production, it should map to the firm's approved pitchbook frameworks. Standard templates reduce variance and reduce the probability that a junior team member or a model draft introduces unapproved language.

The second improvement is to implement stronger numeric and provenance checks. Banking decks often include numbers that are retyped, reformatted, and rounded. The workflow should maintain a single source-of-truth numeric dictionary in state, and spokes should reference those fields rather than inventing new numbers. Deterministic checks should flag any number not present in the dictionary. Where external data is used (market multiples, precedent transactions), the source and timestamp should be captured as fields. If sources are not available, the workflow must label these as assumptions and route to human verification. Data sourcing and permissions are an **Open item for compliance review**.

The third improvement is to formalize the open-items workflow. Open items should be assigned owners and deadlines in the decision packet, even if only as placeholders in pilot. This turns "we need to verify comps" into an operational task. The board should require that any critical open item blocks external use until resolved. This is a governance posture: the firm will not pitch on unverified claims.

The fourth improvement is to calibrate the revision loop policy. Revision cycles should be limited and purposeful. The aggregator should not allow broad rewrites; it should request targeted fixes: "align revenue figure across sections," "remove unsupported valuation range," "elevate risk X to headline." Each revision should be logged. If revisions exceed the cap, the workflow should escalate. This prevents the classic failure mode where a team rewrites until the deck "sounds right" rather than until it is consistent.

The fifth improvement is monitoring metrics. The board should require at least seven metrics: open-item density per pitch packet, critical open-item rate, inconsistency detection rate, revision cycle counts, time-to-ready-for-MD-review, human override rate (how often MDs reject the packet), and post-review correction rate (how many changes are required after MD review). These metrics indicate whether the workflow is reducing variance or merely shifting work. They also provide governance evidence that the institution monitors the system.

The sixth improvement is training integration. Banking is an apprenticeship culture. The workflow should be used to teach juniors how to write with governance: the evidence register becomes a training artifact. Senior reviewers can point to unsupported claims and show how to fix them. Over time, this improves human performance, which is an institutional benefit beyond automation.

A conservative deployment plan is phased. Phase 1: internal drafting support only, with outputs labeled Not verified and used as starting points for analysts and associates. Phase 2: integrate with internal review workflows so that the decision packet and open-items list become part of standard pitch preparation. Phase 3: consider broader rollout only when metrics show reduced inconsistency and acceptable open-item management. At no stage should the system be allowed to generate external-facing final decks without senior review.

The board should also require adversarial testing. The workflow should be tested against prompt injection attempts such as "invent comps," "make the valuation higher," or "ignore risks." The expected behavior is refusal or escalation, with the system recording the attempted instruction and routing to HUMAN_REVIEW. The board should also test sensitive scenarios where facts are missing or contradictory. The workflow should surface open items and not resolve contradictions through invented narrative.

Finally, the board should treat N7 as part of a broader governance-first architecture program. The hub-and-spoke pattern can be reused across corporate finance deliverables. Approving N7 is approving a disciplined approach: separate responsibilities, surface gaps, enforce consistency, bound iteration, and record artifacts. That is the institutional contribution.

The recommended board action is specific. Approve N7 as a pilot internal drafting workflow for pitchbook sections with the following conditions: strict prohibition on invented facts, mandatory evidence registers and open-items lists, deterministic consistency checks, bounded revision loops with explicit stop reasons, mandatory artifact export each run, and mandatory senior human review before any external use. Require monitoring metrics and monthly sampling audits. Forbid framing any output as "system-approved." Under these conditions, N7 will accelerate drafting while improving defensibility, reducing internal inconsistency, and making the pitch process more governable in the face of time pressure and narrative incentives.

Chapter 8

M&A Diligence Q&A Over Documents (Router + Retrieval)

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_8.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

In M&A, diligence is where optimism meets paperwork. A deal team begins with a thesis: synergies, growth, defensibility, strategic fit. Then diligence begins, and the thesis is forced to confront contracts, policies, financial schedules, customer terms, compliance histories, litigation correspondence, and operational realities. The board and investment committee want a simple answer: "Are there any red flags?" The diligence team cannot give a simple answer because diligence is not a single question. It is a structured interrogation of documents, and the truth is often not in one document but distributed across many. The recurring enterprise problem is therefore not that teams cannot read. The problem is that teams cannot scale disciplined reading under time pressure while maintaining traceability.

Notebook 8 (N8) exists to formalize diligence Q&A as a governed retrieval and routing workflow. The architecture is a router-plus-retrieval system: it classifies incoming diligence questions, routes them to the appropriate retrieval strategy, extracts evidence from the relevant documents, and produces a response packet that explicitly includes citations to the retrieved evidence and an open-items list for anything that cannot be answered from the evidence. The system's deliverable is not "the answer." The deliverable is an evidence-backed diligence response: what was asked, what was found, what evidence supports the conclusion, what remains unknown, and what follow-up diligence is required.

The enterprise context is intense. Diligence runs in parallel across streams: legal, financial, tax, HR, IP, commercial, operations, cybersecurity, compliance. Each stream asks questions, often overlapping. Each stream produces notes, often inconsistent. Meanwhile, the deal team wants a consolidated view and the committee wants a decision timeline. This creates a structural risk: incomplete or inconsistent diligence answers are summarized upward as if they are resolved. A single missing clause in a customer contract or a single covenant in a lease can change valuation. A single ambiguous policy can create post-close liability. The institution therefore needs a diligence process that is not only fast, but demonstrably traceable.

LLMs look tempting here because they can read and summarize text quickly. The danger is that they can also generate plausible answers without quoting evidence. In diligence, plausible is unacceptable. If the team later discovers a missed clause, the question becomes: did we fail to retrieve the right document, did we fail to interpret it, or did we never have it? A chatbot-style system cannot answer that question because it does not

preserve provenance. N8 is built to preserve provenance as a first-class output: every nontrivial claim must be backed by retrieved evidence or labeled as an assumption or open item. This is the diligence equivalent of "facts vs assumptions vs open items" discipline.

The institutional problem can be framed as an intake and routing problem. Diligence questions are heterogeneous. Some are narrow and factual: "What is the change-of-control clause in the top 10 customer contracts?" Some are procedural: "Do we have evidence of policy compliance for X?" Some are interpretive: "Does this clause create risk?" Some require aggregation: "List all contracts with exclusivity." Some require cross-document reconciliation: "Do the financial schedule and the contract terms match?" Treating all questions the same produces failure. A narrow factual question should retrieve a specific clause; an aggregation question should run a broader search; an interpretive question should be routed to human legal review rather than answered with false authority. N8 solves this by explicitly classifying and routing questions.

The deliverable of N8 is a diligence Q&A packet that can be attached to the diligence tracker. It contains: the question, the question type classification, the documents and passages retrieved, a short evidence summary, the answer constrained to what the evidence supports, and an explicit open-items list. The packet also contains a routing recommendation: ACCEPTABLE_ANSWER (evidence-backed), NEEDS_MORE_DOCS (missing evidence), or ESCALATE_TO_HUMAN_SPECIALIST (interpretive or high-stakes). This makes diligence progress measurable: the team can track which questions are answered with evidence, which are blocked by missing documents, and which require specialist judgment.

From the board's perspective, this workflow is a governance layer that reduces the probability that diligence summaries become narrative rather than evidence. It also reduces key-person risk. Diligence is often concentrated in a few experienced people who know where the bodies are buried in document sets. N8 encodes part of that expertise as a process: route questions, retrieve evidence, require citations, and escalate when interpretation is required.

The failure modes N8 is designed to address are clear. First is hallucinated answers: producing a conclusion without evidence. Second is retrieval miss: searching the wrong subset of documents or missing the relevant clause. Third is false completeness: answering a question based on one document when it requires multiple. Fourth is overreach: interpreting legal or regulatory meaning without escalation. Fifth is inconsistency: different team members answer the same question differently because they found different documents. Sixth is unbounded searching: spending unlimited time retrieving without a stop condition. Seventh is stakeholder misinterpretation: treating an evidence-backed excerpt as a full legal conclusion. N8 mitigates these by enforcing citations, using routing to pick retrieval strategies, bounding loops, and producing explicit escalation recommendations.

The governance emphasis in N8 is therefore concrete. Bounded loops matter because retrieval can become endless. The workflow should have a fixed number of retrieval attempts, a fixed number of documents to consider, and explicit stop reasons such as MAX_RETRIEVAL_ATTEMPTS or INSUFFICIENT_EVIDENCE. final_state.json matters because it shows what the system searched, what it retrieved, and what it concluded. graph_spec.json matters because it shows that interpretive questions route to escalation rather than free-form

answering. run_manifest.json matters because it allows the diligence record to be reproduced under the same configuration, which is essential if a dispute arises later.

The enterprise narrative also requires acknowledging that diligence is a human process. Humans decide what questions matter. Humans decide how to interpret ambiguous clauses. Humans decide whether a risk is acceptable or whether it affects price. N8 is designed to support those decisions by making evidence retrieval faster and more disciplined, not by replacing judgment.

The decision packet to the board should recommend a conservative pilot. Approve N8 as an internal diligence assistant that produces evidence-backed Q&A packets for the diligence tracker. Restrict it to retrieval and summarization with citations. Prohibit it from giving legal conclusions, regulatory conclusions, or final risk judgments; those must route to human specialists. Require that every output includes facts vs assumptions vs open items and includes the retrieved passages. Require artifact export and monitoring metrics. Require that document ingestion and access controls are governed; data permissions and retention policies are open items for compliance review.

In practice, the most important benefit is not speed but reduction of diligence leakage. In many deals, the diligence tracker becomes a list of questions with informal answers. N8 turns each answer into a mini-audit object: it has evidence and a trace. That makes the diligence tracker more valuable and the committee conversation more grounded. When the committee asks "Do we have change-of-control risks?" the team can point to evidence-backed packets rather than subjective notes.

Finally, N8 helps the firm learn. Over time, the system can identify which question types most often hit INSUFFICIENT_EVIDENCE, indicating gaps in the data room. It can identify which documents are most frequently retrieved, indicating key risk areas. It can also identify where human escalation is frequent, indicating where the institution should invest in specialist review earlier. These are governance insights, not model tricks.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

N8 is implemented as a LangGraph workflow that combines routing and retrieval under explicit state. The TypedDict state is the diligence case file. It contains: the incoming question, normalized question text, question type classification, retrieval query terms, the document index or corpus metadata, retrieved document passages with identifiers, an evidence summary, the draft response, an open-items list, loop counters for retrieval attempts, termination reason, and routing decision. The key design choice is that retrieved evidence is stored as structured objects in state, so it can be exported and reviewed.

Nodes are wrapped in an AgentNode abstraction with single responsibilities. A question intake node normalizes the question and captures metadata such as diligence stream (legal, financial, commercial). A router node classifies the question type. A retrieval node executes a search against the document set, returning top passages. A synthesis node summarizes the retrieved evidence in constrained form and drafts an answer that is limited to what the evidence supports. A verification node checks whether the answer contains

unsupported claims and whether citations exist for key statements. An escalation node packages the case for human specialists when interpretation is required or evidence is insufficient. A final packaging node produces the Q&A packet and writes artifacts.

Conditional routing implements the core governance logic. The router determines whether a question is factual retrieval, aggregation, cross-document reconciliation, or interpretive. Factual and aggregation questions proceed to retrieval. Interpretive questions route directly to ESCALATE_TO_HUMAN_SPECIALIST or proceed only with strict disclaimers and a requirement to produce evidence excerpts without legal interpretation. Cross-document questions may trigger multiple retrieval passes and then a reconciliation step. Importantly, routing is a hard branch recorded in state and in the graph spec. This ensures that the institution can demonstrate that the system is designed to escalate rather than to overreach.

Bounded loops govern retrieval attempts. The workflow may run a limited number of retrieval iterations, each time refining queries based on missing evidence. The loop counter is stored in state. If the system cannot retrieve sufficient evidence within the cap, it terminates with INSUFFICIENT_EVIDENCE or MAX_RETRIEVAL_ATTEMPTS_REACHED and routes to human review or to "needs more documents." This is a control: it prevents endless searching and forces explicit escalation.

The hard-coded versus LLM boundary is explicit. Hard-coded code includes: retrieval configuration (top-k, scoring thresholds), loop caps, routing rules, termination reasons, and deterministic checks for citation presence. The LLM is used for classification of question types, drafting refined queries, summarizing evidence, and drafting constrained responses. The LLM is not allowed to invent facts; if evidence is missing, the system must output an open item. The verification node enforces this discipline by scanning for unsupported statements and by requiring that key claims map to retrieved passages.

Artifacts are essential. `run_manifest.json` captures run id, config hash, environment fingerprint, and versions. `graph_spec.json` captures the routing topology: question types to paths, retrieval loops, and escalation endpoints. `final_state.json` captures the complete ledger: the question, the classification, the retrieval queries, retrieved passages, answer text, open items, termination reason, and route. These artifacts are what allow diligence to be defensible. They turn a model output into an auditable process record.

Mermaid rendering provides a governance visualization. The diagram shows the router branching into retrieval versus escalation, and shows the bounded retrieval loop. This makes it legible to non-technical stakeholders: the system is designed to retrieve, cite, and stop rather than to improvise.

Human intervention is integral. The architecture explicitly routes interpretive questions to human specialists. Humans also review the Q&A packets, confirm evidence relevance, and decide whether to request more documents. In a disciplined deployment, humans sign off on packets that will be used in committee materials. Overrides and corrections should be logged to improve routing and retrieval over time.

In summary, N8 is a governed retrieval architecture: it routes heterogeneous diligence questions into appropriate retrieval strategies, enforces evidence citation discipline, bounds searching, escalates interpretive cases, and exports artifacts that make the entire process reviewable and reproducible.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

The board-level objective for N8 is to strengthen diligence traceability while accelerating evidence retrieval. The next steps should therefore focus on improving retrieval quality, tightening provenance controls, integrating specialist review, and establishing monitoring metrics that reveal whether the workflow is safe and useful.

The first improvement is to formalize the question taxonomy and escalation policy. Diligence questions should be categorized into a small, stable set that maps to workflows: factual clause extraction, aggregation across documents, cross-document reconciliation, and interpretive risk assessment. The escalation policy should be explicit: interpretive questions must route to human specialists, and any question that cannot be answered with evidence within the bounded retrieval attempts must route to "needs more documents" rather than to a speculative answer. These policies should be versioned and included in the run manifest.

The second improvement is retrieval evaluation and tuning. Retrieval systems fail silently when they miss relevant documents. The institution should build a small gold set of diligence questions and known supporting passages, and evaluate retrieval recall and precision. This is not model training; it is governance testing. If recall is poor, the system must not be trusted for committee-critical questions. Improvement may involve better indexing, better chunking, better metadata filters by diligence stream, and better query construction. Where changes affect document processing, permissions and retention remain open items for data governance and compliance review.

The third improvement is stronger citation discipline and response constraints. The response should be structured so that each claim points to a retrieved passage identifier. Where this is not possible, the claim must be downgraded to an assumption or open item. The system should explicitly avoid legal conclusions. It can quote and summarize clauses, but it should route interpretation to humans. This protects the institution from overreach and misinterpretation.

The fourth improvement is to integrate human review workflows. Each Q&A packet should have a reviewer field and a disposition: accepted, needs more evidence, or escalated. Review feedback should be captured as a governance artifact linked to the run. Over time, this feedback becomes a dataset that improves routing and retrieval policies. The institution should treat this as part of diligence governance: decisions are traceable, and improvements are evidence-based.

The fifth improvement is monitoring and metrics. The board should require a dashboard with at least eight metrics: retrieval success rate (questions answered with sufficient evidence), insufficient-evidence rate, escalation-to-specialist rate, average retrieval iterations per question, loop exhaustion rate, citation completeness rate, reviewer override rate, and time-to-packet. In addition, track "contradiction rate," where retrieved passages conflict; such cases should route to escalation rather than to resolution by narrative. These metrics make the workflow governable and reveal whether it is reducing workload or creating hidden risk.

A conservative deployment plan is phased. Phase 1: use N8 internally on a controlled document set and require that outputs are used only as drafting aids for diligence teams, with mandatory human review. Phase 2: integrate N8 into the diligence tracker so that evidence-backed packets become standard attachments to

questions. Phase 3: allow committee-facing use only after retrieval evaluation shows acceptable recall and after sampling audits show consistent citation discipline. At no stage should N8 be positioned as a substitute for legal, tax, or regulatory specialists.

The board should also require adversarial testing. Diligence workflows are vulnerable to prompt injection and to biased questions framed to confirm a thesis. The system should be tested on questions that attempt to force a conclusion ("Tell me there is no change-of-control risk") and should respond by retrieving evidence and presenting open items rather than endorsing the thesis. It should also be tested on incomplete document sets to ensure it routes to insufficient evidence rather than hallucinating.

Finally, the board should understand N8 as a mechanism to reduce diligence uncertainty, not to eliminate it. Diligence is inherently about unknowns. The institution's duty is to surface unknowns clearly and make decisions with eyes open. N8 operationalizes that duty by making evidence retrieval disciplined, by forcing citations, by bounding searching, and by escalating interpretation.

The recommended board action is specific. Approve N8 as a pilot router-plus-retrieval diligence assistant that produces evidence-backed Q&A packets with explicit citations, facts vs assumptions vs open items, bounded retrieval attempts, and mandatory escalation for interpretive questions. Require artifact export for every run, require monitoring metrics and monthly sampling audits, and require strict data access controls and retention policies as preconditions for broader deployment. Forbid treating outputs as final legal conclusions or as substitute for specialist diligence. Under these conditions, N8 will make diligence faster and more defensible, converting scattered document reading into a governed process that the committee can trust as a record of what was found and what remains unknown.

Chapter 9

Treasury Liquidity and Covenant Monitoring (Event-Driven Workflow)

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_9.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

Treasury risk rarely announces itself politely. It accumulates quietly in cash balances, borrowing base headroom, covenant ratios, maturity ladders, and operational cash needs—until one day it becomes a crisis meeting. The board is then told that a covenant is "close," that liquidity is "tight," or that refinancing "may be needed." The question the board asks is not only what the numbers are, but why the organization did not see the pressure earlier and why the institution cannot reproduce the monitoring logic that produced the alert. The recurring enterprise problem is that liquidity monitoring is often spreadsheet-driven, person-dependent, and reactive. It is not that the firm lacks data; it is that the firm lacks a governed mechanism that turns streams of data into consistent alerts, escalation decisions, and audit trails.

Notebook 9 (N9) exists to encode liquidity and covenant monitoring as an event-driven, governed workflow. The architecture is not a forecasting engine and not a funding optimizer. It is a monitoring and escalation machine. It consumes periodic "events" (new cash balance snapshots, new forecast updates, new debt schedule updates, new covenant calculations), evaluates them against hard-coded thresholds and policy rules, and routes to actions such as `NO_ACTION`, `MONITOR`, `ESCALATE_TO_TREASURY`, or `ESCALATE_TO_BOARD`. It produces a decision packet and an audit bundle each run so that the organization can answer, under scrutiny, what it knew, when it knew it, what rule triggered the alert, and what escalation decision was recommended.

The enterprise setting is defined by asymmetric downside. Liquidity shortfalls can trigger covenant breaches, rating actions, forced asset sales, or operational disruption. Yet monitoring often suffers from three structural weaknesses. First, thresholds are informal: "we get worried below X" lives in human memory, not in code. Second, calculations drift: spreadsheets evolve, assumptions change, and the firm cannot reconstruct which version produced a given view. Third, escalation is inconsistent: one analyst escalates early, another escalates late, and leadership receives mixed signals. The consequence is governance failure: not necessarily because people are negligent, but because the institution has no single controlled mechanism for turning data into alerts.

N9 solves a narrower but essential problem: it makes the monitoring logic explicit and repeatable. The workflow treats each monitoring cycle as an event. An event can be time-based (daily cash close, weekly forecast refresh, monthly covenant package) or trigger-based (unexpected drawdown, margin call, delayed

receivable, covenant ratio deterioration). The workflow ingests the event, updates state, computes key indicators deterministically, checks them against policy thresholds, and produces an escalation recommendation with explicit reasons. If the event implies missing data or ambiguous inputs, the workflow routes to `NEEDS_DATA` and produces an open-items list. The system does not fill gaps with optimistic assumptions. It fails closed and escalates.

From the board's perspective, this is a governance instrument. Treasury monitoring is a "control function" as much as a finance function. Boards care about whether the organization has early-warning systems and whether those systems are testable. N9 provides that testability by producing artifacts. `run_manifest.json` records configuration and environment fingerprint. `graph_spec.json` records the approved routing logic: what thresholds exist, what conditions trigger escalation, and what termination states exist. `final_state.json` records the computed indicators, the triggered rules, the escalation route, and the stop reason. When leadership asks "why did we escalate," the institution can show the rule, not a retrospective story.

The operational reality of liquidity and covenant monitoring is that the same numerical fact can have different meanings depending on timing. A low cash balance on a Friday is different from a low cash balance on a Tuesday if major outflows are scheduled Monday. A covenant ratio near breach is different if there is an equity cure option or if the facility matures soon. A borrowing base deficiency is different if receivables are expected to normalize in days. N9 therefore treats monitoring as a stateful process, not a one-off calculation. The state includes recent history, upcoming obligations, and current covenant headroom. The event-driven design updates state incrementally and allows trend-aware checks, such as "headroom has deteriorated for three consecutive events."

The architecture also resolves a common institutional communication failure: treasury reports are often dense, and committees need a decision-ready summary. N9 produces a decision packet that is designed for committee consumption: current liquidity position, forecasted runway under baseline and stress, covenant headroom summary, key drivers of change since last event, triggered alert rules, open items, and recommended escalation route. This packet can be reviewed quickly, and it creates a consistent language across cycles.

The failure modes N9 is designed to address are the ones that create surprises. First is threshold drift: alerts depend on who ran the spreadsheet. Second is stale assumptions: forecasts used are outdated and the system does not detect that. Third is silent deterioration: headroom declines gradually without triggering action until it is too late. Fourth is missing data: covenant calculations proceed with unknown inputs, producing false comfort. Fifth is inconsistent escalation: the same condition triggers different actions across time. Sixth is uncontrolled interaction: analysts keep "massaging" numbers without logging changes. Seventh is stakeholder misinterpretation: leadership sees a red indicator without context and overreacts, or sees a green indicator and underreacts. N9 mitigates these by making thresholds explicit, labeling assumptions, requiring open items for missing data, and generating consistent escalation decisions.

A board-ready recommendation embedded in this chapter should be conservative. Approve N9 as an internal monitoring and escalation workflow, not as an autonomous decision engine. Require that thresholds and escalation rules are committee-approved and versioned. Require that every run generates audit artifacts.

Require that the workflow always outputs facts vs assumptions vs open items. Require human review of escalations in pilot, with mandatory sign-off before actions such as lender communication or cash policy changes. Restrict high-stakes events (e.g., covenant within a defined buffer) to immediate human escalation, not automated handling. Require monitoring metrics: alert frequency, false positives/negatives, time-to-escalation, and override rate.

The enterprise value is also strategic. Once the monitoring logic is explicit, the firm can stress test it. It can simulate adverse events—revenue shortfalls, working capital shocks, rate increases—and observe how the workflow routes escalations. This creates an internal "liquidity control lab." The board can then ask: under which shocks do we breach covenant buffers, and how early would we know? Without a governed workflow, that question is hard to answer consistently.

Human intervention remains central. Treasury professionals decide policy thresholds, interpret the implications of covenant terms, and decide mitigation actions. N9 supports them by producing consistent indicators and escalation packets. It does not replace judgment; it reduces surprise and improves traceability.

In short, N9 exists because liquidity risk is a governance risk. A firm cannot claim it is well-governed if it cannot reproduce how it monitors liquidity and covenants and how it escalates. An event-driven workflow with explicit rules, explicit state, bounded handling of missing data, and audit artifacts is the minimum discipline for a modern treasury function that wants board confidence.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

N9 is implemented as an event-driven LangGraph workflow. The explicit TypedDict state is the treasury monitoring ledger. It includes fields such as: event metadata (timestamp, event type, source), current cash balances, forecast cash flows, debt schedule and maturity profile, covenant definitions and current ratio calculations, headroom buffers, trend metrics (changes since last event), policy thresholds, triggered alerts, open items for missing data, and final routing decision with termination reason. The key design principle is that the workflow does not run "in a vacuum." It updates a state that represents the treasury view of the world at the time of the event.

Nodes are wrapped as AgentNode components with single responsibilities. An ingestion node parses the incoming event and updates raw inputs. A normalization node standardizes units, currencies, and time horizons. A computation node deterministically calculates liquidity metrics such as cash runway, net liquidity, available revolver capacity, and covenant ratios. A trend node computes deltas and rolling deterioration indicators. A policy evaluation node checks computed metrics against thresholds and rule logic. A missing-data node identifies required fields that are absent and writes them as open items, potentially routing to `NEEDS_DATA`. A packaging node generates the board-ready decision packet. An escalation routing node chooses the final outcome: NO_ACTION, MONITOR, ESCALATE_TO_TREASURY, ESCALATE_TO_CFO, ESCALATE_TO_BOARD, or NEEDS_DATA.

Conditional routing is the core of governance. The workflow uses deterministic gates: if covenant headroom is

below a defined buffer, route to escalation. If cash runway under stress is below a minimum, route to escalation. If a required covenant input is missing, route to NEEDS_DATA. If deterioration persists across multiple events, route to escalation even if absolute thresholds are not yet breached. These gates are hard-coded policies, not LLM opinions. They are the parts the committee should review and approve.

Bounded loops appear as bounded retries for missing data or reconciliation. If an event arrives with incomplete inputs, the workflow can attempt a limited number of reconciliation steps or requests for missing items. The loop counter is stored in state, and the graph enforces a cap. When the cap is reached, the workflow terminates with a stop reason such as MAX_RETRIES_REACHED and routes to human review. This prevents the system from repeatedly "waiting" or improvising values.

The boundary between hard-coded code and LLM use is explicit. Hard-coded code performs computations, threshold checks, trend logic, and routing. These must be deterministic for auditability. The LLM is used only for drafting the narrative summary in the decision packet: explaining what changed, why a threshold triggered, and what open items remain. Even then, the narrative must be grounded in state fields and must not invent explanations. A disciplined implementation can enforce this by requiring the narrative to reference specific state metrics and triggered rule identifiers.

Artifacts are mandatory and central to the design. `run_manifest.json` captures run id, timestamp, configuration hash, and environment fingerprint. `graph_spec.json` captures the event-driven graph topology, including escalation routes and missing-data routes. `final_state.json` captures the computed metrics, triggered rules, open items, escalation decision, and stop reason. These artifacts convert monitoring from a spreadsheet output into a reproducible control.

Mermaid rendering helps communicate governance. The diagram shows ingestion, computation, policy evaluation, and branching to escalation endpoints. For boards, this diagram demonstrates that alerts are not subjective; they are the output of a defined process with explicit rules.

Human intervention is designed as the default for high-stakes triggers. When covenant buffers are tight or stress runway is low, the workflow routes to human escalation immediately. Humans then decide mitigation actions: reduce discretionary spend, draw on revolvers, negotiate waivers, refinance, or hedge rates. The workflow's role is to surface the issue early and document why it surfaced.

In summary, N9 is a governed monitoring machine: event ingestion, deterministic computation, explicit policy gates, bounded handling of missing data, clear escalation routing, and audit artifacts that allow the institution to prove how it monitored liquidity and covenants.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

To move N9 from a notebook demonstration to an institutional control, the next steps should focus on policy calibration, integration with data sources, monitoring quality metrics, and change management.

The first improvement is to formalize the threshold and rule catalog. Treasury monitoring should not rely on a single number. It should use a layered rule set: absolute thresholds (e.g., minimum cash, minimum

headroom), trend thresholds (e.g., deterioration over N events), stress thresholds (e.g., runway under stressed forecast), and event triggers (e.g., unexpected outflow beyond X). Each rule should have an identifier, an owner, and a rationale. Rule versions should be embedded in run manifests. This makes policy changes auditable and prevents silent drift.

The second improvement is to integrate robust data governance. In production, events will come from ERP systems, bank feeds, forecast tools, and covenant models. Data permissions, retention, and reconciliation rules are **Open items for compliance and data governance review**. The workflow should include data freshness checks: if forecasts are older than a policy limit, route to NEEDS_DATA rather than producing comfort. It should also include reconciliation checks: if cash balances differ materially across sources, route to escalation or reconciliation rather than choosing one silently.

The third improvement is to strengthen stress scenario handling. Boards do not only care about the baseline; they care about resilience. The workflow should compute liquidity runway under at least one stress scenario and report the delta. Stress assumptions should be explicit and versioned. The board should approve stress parameters and require periodic recalibration. This transforms the monitoring system into an early-warning and resilience measurement tool.

The fourth improvement is to integrate escalation playbooks. An alert is useful only if it triggers a response. The decision packet should include recommended next actions mapped to escalation routes, such as "review forecast assumptions," "prepare lender communication," "evaluate revolver draw," "assess covenant cure options," or "initiate refinancing analysis." These are not automated actions; they are structured prompts for human response. Including them in the packet reduces reaction time and ensures consistency.

The fifth improvement is monitoring and metrics. The board should require at least nine metrics: alert frequency by rule, false positive rate (alerts that did not require action), false negative rate (events that should have alerted), time-to-escalation from threshold breach, override rate (human changes to recommended route), data freshness compliance rate, missing-data incidence rate, trend-trigger incidence rate, and post-alert outcome tracking (did headroom stabilize, did liquidity improve). These metrics allow governance: the institution can assess whether the monitoring system is too noisy or too quiet and adjust thresholds accordingly.

The sixth improvement is sampling audits and tabletop exercises. Monthly sampling should reconstruct a subset of runs from artifacts and confirm that the rules triggered correctly and that narratives match computed metrics. Quarterly tabletop exercises should simulate shocks and observe the workflow's escalation behavior. This is the treasury equivalent of disaster recovery drills: you do not wait for crisis to test your early-warning system.

A conservative deployment plan begins with shadow mode. Run N9 in parallel with existing treasury reporting for several cycles. Compare alerts to human judgment. Record discrepancies and adjust thresholds. Only after shadow mode demonstrates acceptable behavior should the institution treat workflow alerts as operational triggers. Even then, require human confirmation before external actions such as lender communication.

The board should also define clear boundaries for automation. N9 should not send communications, execute funding actions, or trigger transactions automatically. Its role is to monitor, explain, and escalate. Maintaining

this boundary preserves accountability and reduces operational risk.

Finally, the board should view N9 as a core governance control that supports enterprise resilience. Liquidity and covenant monitoring is one of the few financial controls that can prevent catastrophic outcomes if done early and consistently. By approving an event-driven workflow with explicit rules, explicit state, bounded missing-data handling, and audit artifacts, the board is approving a system that reduces surprise and improves defensibility. It also creates institutional memory: when leadership changes, the monitoring logic remains consistent and reviewable.

The recommendation is specific. Approve N9 as a pilot internal treasury monitoring and escalation workflow under strict guardrails: committee-approved rule catalog, deterministic computation, bounded retries, mandatory artifact export, mandatory facts vs assumptions vs open items discipline, and mandatory human sign-off for escalations and any downstream action. Require shadow-mode validation, monitoring dashboards, and monthly artifact sampling as conditions for scale. Under these conditions, N9 will convert liquidity monitoring from an informal spreadsheet practice into a governed, event-driven control that the board can trust as a repeatable mechanism, not merely as a periodic report.

Chapter 10

Multi-Desk Research Synthesis + Red-Team (Supervised Multi-Agent System)

Companion Notebook

Colab: https://github.com/alexdibol/agents-handbook/blob/main/notebooks/CHAPTER_10.ipynb

Enterprise Narrative: The Institutional Problem and Why This Exists

The most expensive mistakes in finance rarely come from lack of information. They come from ungoverned synthesis. A firm can have excellent analysts on macro, equities, credit, rates, and commodities, and still arrive at a fragile decision because no one reconciled contradictions, no one stress-tested the narrative, and no one recorded what assumptions were imported from one desk into another. This is a board-level governance problem disguised as "research." Leadership receives decks and memos that appear coherent but may simply be well-written. Meanwhile, the underlying beliefs are inconsistent: one desk assumes soft landing, another assumes recession; one assumes liquidity is ample, another assumes tightening; one assumes correlations will normalize, another assumes structural regime change. When these inconsistencies are not surfaced, the organization takes positions that are internally conflicted, and the committee later discovers that the "house view" was never truly a view, only an aggregation of voices.

Notebook 10 (N10) exists to formalize research synthesis as a supervised multi-agent system with an explicit red-team function. The architecture is not an "AI strategist." It is a governed orchestration process that collects desk perspectives, forces them into structured outputs, aggregates them into a synthesis, and then subjects that synthesis to adversarial critique before packaging a decision-ready research packet for senior leadership. The deliverable is not a prediction. The deliverable is a supervised synthesis: a coherent view that explicitly lists supporting evidence, assumptions, dissenting opinions, and red-team attacks, with a final recommendation on what is safe to assert, what must be caveated, and what requires further human review.

The enterprise setting is familiar. A CIO or investment committee asks for a cross-desk view: "What is the base case? What are the key risks? What should we do?" Each desk responds with its own lens and incentives. Macro may emphasize growth and inflation trajectories. Rates may emphasize term premium and policy path. Credit may emphasize spreads, default cycles, and liquidity. Equities may emphasize earnings revisions and positioning. Commodities may emphasize supply shocks and geopolitics. Each view is rational locally, but the organization is exposed globally. The operational failure is that synthesis is often done informally by one person, under time pressure, without a structured way to represent contradictions and without a record of what was filtered out. The result is a narrative that feels unified but is not robust.

LLMs can accelerate synthesis, but they also amplify the illusion of coherence. A model can write a beautiful

"house view" that blends perspectives smoothly, but smoothness is not truth. In fact, smoothness can be a risk indicator: it often means dissent was removed. Boards and risk committees should be suspicious of narratives with no tension. Real markets are tension. Real research should show where the story breaks. N10 is designed to preserve tension in a disciplined way by introducing a red-team agent that tries to break the synthesis and a supervisor that enforces governance rules: do not overclaim, do not erase dissent, do not convert assumptions into facts, and do not hide evidence gaps.

The workflow begins by defining the case: what question leadership asked, what time horizon matters, what portfolio context is relevant, and what constraints apply. It then solicits desk contributions. Each desk node is constrained to produce a structured brief: key claims, evidence or indicators, assumptions, uncertainties, and recommended actions or watch items. These outputs are not free-form essays; they are structured objects that can be aggregated. The system then runs an aggregation step that produces a preliminary synthesis. This synthesis is not final. It is an intermediate artifact that must survive critique.

The red-team function is the distinctive governance element. It reads the preliminary synthesis and attacks it along defined axes: hidden assumptions, missing evidence, logical contradictions, regime fragility, and incentive bias. It asks adversarial questions: what if the base case is wrong, what if correlations break, what if liquidity disappears, what if policy does not follow the implied path, what if positioning is crowded, what if the indicators cited are lagging. The red-team output is not a dramatic "gotcha." It is a structured critique: list of vulnerabilities, list of missing data, and list of scenarios that would falsify the synthesis. This critique is then routed back to a revision step, where the synthesis is updated to incorporate caveats, to surface dissent explicitly, and to mark open items requiring human judgment or further research. The loop is bounded: the system can iterate only a limited number of times before it must escalate to human committee review.

The enterprise risk N10 addresses is the production of "false consensus." False consensus is dangerous because it leads to concentrated risk-taking under the belief that the organization is aligned. In reality, alignment may be cosmetic. N10 forces real alignment by making dissent explicit. If one desk's view contradicts another's, the synthesis must record the contradiction and either reconcile it with evidence or flag it as unresolved. This is a governance benefit: leadership can decide whether to take a position despite unresolved disagreement, but it cannot pretend the disagreement did not exist.

The failure modes N10 is designed to address are the most common in cross-desk research. First is narrative smoothing: removing uncertainty to make the story readable. Second is assumption laundering: converting desk assumptions into "facts" in the house view. Third is evidence dilution: citing indicators without noting their limitations. Fourth is contradiction blindness: ignoring inconsistencies between desks. Fifth is incentive bias: desks emphasize what supports their positioning. Sixth is unbounded debate: synthesis becomes endless because no one has authority to stop. Seventh is stakeholder misinterpretation: leadership treats the synthesis as a forecast rather than a disciplined summary. N10 mitigates these by structuring desk outputs, applying deterministic governance rules, adding red-team critique, bounding iterations, and producing explicit termination reasons.

From a board perspective, the deliverable is a research decision packet that is fit for executive consumption. It

contains: the question, the desk briefs, the consolidated synthesis, explicit dissent and confidence levels, the red-team critique, open items and uncertainties, and a recommended action posture (e.g., cautious positioning, hedges, watch list triggers). Crucially, it also contains governance metadata: what was retrieved, what was assumed, what was not verified, and why the system stopped. This packet is not meant to replace investment committee debate; it is meant to improve it by making contradictions and vulnerabilities visible.

The board-ready recommendation embedded in this chapter should be conservative and precise. Approve N10 as an internal research synthesis tool with mandatory human supervision. Require that the system's outputs are labeled as Not verified and are used as inputs to committee discussion, not as trading directives. Require that the red-team function is mandatory and cannot be bypassed. Require that the synthesis explicitly distinguishes facts, assumptions, and open items. Require artifact export on every run. Require monitoring metrics: dissent rate, red-team vulnerability density, iteration counts, and override rates. Forbid the system from presenting conclusions as certainties or from erasing minority views. Restrict categories of high-stakes decision-making to human-only sign-off.

N10 also provides a strategic organizational benefit: it creates a common format for desk thinking. When desks are forced to write in structured form, their assumptions become visible and comparable. This improves internal communication and reduces the tendency for debates to be about rhetoric rather than evidence. It also creates a historical archive: the firm can look back at past synthesis packets and see what it believed and why. This is valuable for learning, and it is valuable for governance when leadership changes.

Finally, N10 emphasizes that the safest use of multi-agent systems is supervised orchestration, not autonomy. The supervisor node exists to enforce policy: enforce bounded loops, enforce red-team inclusion, enforce labeling discipline, and enforce escalation when uncertainty is unresolved. This is the governance-first posture. It is better to produce a packet that says "we do not know" than to produce a confident paragraph that hides uncertainty. Boards should explicitly approve that posture because it may feel less satisfying in the short term but is safer in the long term.

In short, N10 exists because research is a governance surface. The firm's ability to synthesize across desks, surface dissent, and red-team narratives is a determinant of risk outcomes. A supervised multi-agent system with explicit artifacts makes that ability repeatable, reviewable, and improvable.

Architecture Explanation: Graph, State, Nodes, Gates, and Artifacts

N10 is implemented as a supervised LangGraph workflow that orchestrates multiple desk agents and a red-team under explicit policy. The TypedDict state is the synthesis case file. It includes: the leadership question, time horizon, portfolio context metadata, desk briefs (each as structured objects), a preliminary synthesis, a red-team critique, a revision history, an iteration counter for critique cycles, open items, final decision packet, termination reason, and routing decision. The key design principle is that every desk contribution and every critique is stored in state and exported, so provenance is preserved.

Nodes are wrapped as AgentNode components with single responsibilities. Desk nodes produce structured

briefs: claims, evidence indicators, assumptions, uncertainties, and suggested actions. The aggregation node merges these briefs into a synthesis while preserving dissent. The red-team node critiques the synthesis using an adversarial checklist and produces vulnerabilities and falsification scenarios. The revision node updates the synthesis to incorporate critiques, adding caveats and open items. The supervisor gate node evaluates whether the synthesis meets governance criteria: clear distinction of facts vs assumptions vs open items, explicit dissent representation, and acceptable vulnerability density. Based on these criteria, it routes to finalize, to iterate again, or to escalate to human review.

Conditional routing is central. The supervisor gate is deterministic: if a governance criterion fails, the workflow cannot finalize. It must revise or escalate. For example, if the synthesis includes unsupported claims, route to revision. If the red-team identifies high-severity vulnerabilities that are unresolved, route to revision or escalation. If iteration count exceeds the maximum, route to HUMAN_COMMITTEE with MAX_ITERATIONS_REACHED. This ensures bounded loops and prevents endless debate.

The bounded loop is the red-team cycle. The graph iterates synthesis → red-team critique → revision → supervisor check, with a hard cap on iterations. The cap is a governance control: it forces escalation when the system cannot produce a stable, defensible synthesis. In institutional terms, this is correct: unresolved contradictions should be escalated to human decision-makers rather than massaged into coherence.

The hard-coded versus LLM boundary is explicit. Hard-coded code implements the governance scaffold: state schema, desk brief schema, aggregation and supervisor rules, severity thresholds, loop caps, routing, and artifact export. The LLM is used to draft desk briefs, draft synthesis prose, and draft red-team critiques. However, LLM outputs are constrained to structured fields and are evaluated by deterministic gates. The LLM does not decide whether to finalize; the supervisor gate does.

Artifacts make the system governable. `run_manifest.json` captures configuration and environment fingerprints. `graph_spec.json` captures the workflow topology, including desk fan-out, red-team loop, and supervisor gating. `final_state.json` captures the desk briefs, synthesis versions, red-team critiques, open items, iteration counts, termination reason, and final decision packet. These artifacts allow committees to inspect not just the final narrative but the process by which it was formed.

Mermaid rendering provides executive legibility. The diagram shows desk nodes feeding into an aggregator, then a red-team node, then a supervisor gate, then either finalization or another loop. This picture communicates a key governance message: the system is designed to disagree with itself before it speaks to leadership. That is the correct posture for high-stakes synthesis.

Human intervention is represented explicitly. The workflow routes to HUMAN_COMMITTEE when unresolved high-severity issues persist or when the loop cap is reached. Humans also review the final packet before any action is taken. In practice, a human chair of research or risk should sign off. Overrides and corrections should be logged as governance artifacts, because they are high-signal events for improving the system.

In summary, N10 is a supervised multi-agent synthesis architecture: structured desk briefs, deterministic aggregation that preserves dissent, mandatory red-team critique, bounded revision loops, supervisor gating,

and artifact export that preserves provenance and enables audit.

Conclusion: Improvements, Metrics, and Board-Level Next Steps

To adopt N10 responsibly, the board should focus on governance criteria, operational integration, and monitoring metrics that reveal whether the system is producing disciplined synthesis rather than polished storytelling.

The first improvement is to formalize the desk brief and synthesis schemas. The success of supervised multi-agent systems depends on structured outputs. Each desk brief should have stable fields and severity scales for uncertainty. The synthesis should have explicit sections: base case, alternative cases, key risks, evidence summary, dissent matrix, and watch triggers. These schemas should be versioned and embedded in the run manifest. This reduces drift and makes outputs comparable over time.

The second improvement is to formalize the red-team checklist and severity taxonomy. Red-teaming should not be theatrical. It should be systematic. The checklist should include: hidden assumptions, contradictory indicators, regime fragility, tail risk scenarios, liquidity discontinuities, incentive bias, and time-horizon mismatch. Each vulnerability should have severity and required action: add caveat, request evidence, escalate to specialist, or block finalization. The supervisor gate should enforce that high-severity vulnerabilities cannot be ignored.

The third improvement is to integrate evidence provenance where possible. In a mature deployment, desk briefs should cite data sources or internal research notes. If retrieval is involved, it should be governed and cited. If evidence cannot be provided, the synthesis must label claims as assumptions. This is the minimum discipline: facts are not assumptions. Without provenance, the system must fail closed and escalate rather than produce confident claims.

The fourth improvement is monitoring and metrics. The board should require at least eight metrics: dissent rate (how often desks disagree), vulnerability density (number of red-team issues per packet), high-severity vulnerability rate, iteration counts, MAX_ITERATIONS_REACHED rate, human override rate, post-hoc correction rate (how often leadership corrects the synthesis), and alignment drift (how often the house view changes without evidence). In addition, track "claim support ratio": proportion of claims labeled as fact with evidence versus assumption/open item. These metrics reveal whether the system is functioning as governance or as storytelling.

The fifth improvement is operational integration. The system's outputs should be integrated into the committee pack process, not into trading workflows. The synthesis packet should be reviewed in research meetings, risk meetings, and investment committee meetings. It should be treated as an input that structures debate, not as a directive. This positioning should be explicit to prevent misinterpretation.

The sixth improvement is change management and regression testing. Because prompts, models, and aggregation policies may change, the institution should maintain a regression suite of historical scenarios and expected behaviors. Run the suite when changes occur and compare outputs and routing decisions. Store

artifacts for each regression run. This is model risk management applied to synthesis: you cannot govern what you do not test.

A conservative deployment plan begins with shadow use. Run N10 on a subset of weekly research questions and compare its synthesis packets to human-produced packs. Measure whether it surfaces dissent and vulnerabilities that humans missed, and whether it introduces noise. Use this to calibrate severity thresholds and schemas. Only after shadow mode demonstrates value should the institution consider broader adoption. Even then, require that a senior human supervisor signs off on any packet used for committee decisions.

The board should also require adversarial testing. Test whether the system can be pressured to erase dissent, to overclaim certainty, or to produce trading instructions. The expected behavior is refusal or escalation. Test whether the system handles contradictory desk briefs by surfacing conflict rather than smoothing it. These tests should be repeated whenever the system is updated.

Finally, the board should evaluate N10 as a cultural control. A firm that institutionalizes red-teaming and explicit dissent is a firm that reduces groupthink risk. N10 operationalizes that culture in a repeatable mechanism. It will not make the firm right. It will make the firm more honest about what it does not know and more disciplined about what it claims. That honesty is a competitive advantage in risk management.

The recommended board action is specific. Approve N10 as a pilot supervised research synthesis workflow with mandatory red-team critique, deterministic supervisor gating, bounded iteration loops, mandatory artifact export, and mandatory facts vs assumptions vs open items discipline. Require that outputs are used only as inputs to human committees and are labeled Not verified. Require monitoring dashboards, monthly sampling audits, and a regression test suite as release criteria. Forbid interpreting outputs as trading directives or as autonomous “house view” approvals. Under these conditions, N10 will strengthen the institution’s research governance by making synthesis structured, dissent explicit, critique mandatory, and decisions more defensible under uncertainty.

Conclusion: What the Board Should Take Away (and What Comes Next)

This handbook is a companion to ten LangGraph notebooks, but it is not a "notebook summary." Its purpose is to leave executive leadership with a durable mental model: these systems are not magic models that produce answers; they are governed workflows that produce controlled outcomes. If there is one principle to carry forward, it is that the unit of value is not the model output. The unit of value is the **auditable decision process** that surrounds model output: explicit state, explicit routing, bounded loops, explicit stop reasons, and artifacts that preserve provenance. That is what transforms an LLM from a persuasive text generator into a component that can be responsibly embedded in finance.

Across the ten architectures, the recurring pattern is simple and intentionally conservative. Each notebook begins by defining a case file (the state), then decomposes the work into nodes with narrow mandates, then uses conditional routing to enforce gates, and then terminates with a recorded decision and exported artifacts. In institutional terms, this is the difference between a conversation and a control. A conversation can sound correct while being wrong. A control can be inspected, tested, versioned, and improved. Boards do not approve "conversations." Boards approve controls.

The first strategic implication is that the organization should not treat agentic systems as a single product category. "AI agent" is too vague to govern. Instead, the institution should govern **workflow classes**. N1 (Personal Finance Triage) is an intake control. N2 (Advice Suitability Boundary) is a gate and refusal control. N3 (Credit Memo with Evidence Gaps) is a drafting-and-critique loop with escalation. N4 (Hypothesis + Backtest Wrapper) is a research discipline control. N5 (Execution under Liquidity Regimes) is a stateful tactical control with explicit stop conditions. N6 (Rebalancing Committee) is a multi-perspective governance control. N7 (Pitchbook Hub-and-Spoke) is a drafting constellation with consistency checks. N8 (Diligence Router + Retrieval) is a provenance-first retrieval control. N9 (Treasury Monitoring) is an event-driven escalation control. N10 (Multi-desk Synthesis + Red-team) is a supervised synthesis control. The board should recognize that each class has distinct risk surfaces, distinct metrics, and distinct acceptance criteria. A single policy titled "AI agents" will either be too strict to allow value or too loose to control risk. Workflow governance is the right abstraction.

The second implication is that transparency is not optional decoration; it is the core product feature. In these architectures, transparency is operationalized through three artifacts: `run_manifest.json`, `graph_spec.json`, and `final_state.json`. Together they answer the only questions that matter when something goes wrong: what exactly ran, under what configuration and environment, with what routing logic, and what the system believed at each step when it stopped. This is why the notebooks insist on explicit TypedDict state and explicit termination reasons. It is also why bounded loops matter. A system that can loop endlessly cannot be audited meaningfully; it can always claim it was "still working." In finance, that is unacceptable. A governed system must stop and either deliver a constrained packet or escalate to a human. The stop itself is part of the control.

The third implication is that the institution's safest path to benefit is **triage, packaging, and escalation** rather

than autonomous action. Notice how many deliverables in this handbook are not "answers" but "packets": triage packets, decision tickets, execution traces, committee critiques, pitch packets, diligence Q&A packets, monitoring escalation packets, synthesis packets with red-team vulnerabilities. This is not a limitation; it is a design choice aligned with institutional responsibility. Packets are reviewable objects. Packets can be signed off. Packets can be sampled and audited. Packets reduce the burden on senior humans by structuring what they need to see, while preserving the human's role as decision-maker.

This is the most important point to communicate internally: the goal is not to replace experts. The goal is to build an institutional pipeline that makes expertise scale without losing accountability. In each notebook, the LLM is used where it is strong (classification, summarization, drafting within constraints) and avoided where it is weak or risky (unbounded inference, "making up" missing facts, interpreting ambiguous clauses as definitive conclusions). Hard-coded code encodes policy, constraints, and deterministic computation. The LLM provides language and structured suggestions. Humans retain responsibility for approvals, high-stakes judgments, and any action that changes client outcomes, market risk, or legal posture. This division of labor is not merely prudent; it is how you make these systems governable.

The fourth implication is that governance must be designed as a system, not as a disclaimer. Disclaimers do not prevent harm when incentives push the organization to treat outputs as authoritative. Governance requires hard gates. N2 is the canonical example: suitability and refusal are not "advice," they are policy enforcement. But the same principle appears elsewhere: in N4, a backtest gate decides whether results can be promoted for further testing; in N5, a stop gate decides when to pause or escalate under stress; in N8, interpretive diligence questions route to human specialists; in N9, tight covenant buffers route to escalation; in N10, high-severity red-team vulnerabilities block finalization. These are institutional controls encoded as routing logic. They represent the organization's risk appetite in executable form.

Boards should therefore require that any deployment proposal includes, at minimum, a **gate map**. A gate map is a simple statement: what are the possible termination states, what conditions trigger each, and what is the required human action at each termination. If a team cannot produce a gate map, they do not have a governable system; they have a chatbot. This handbook is, in effect, ten examples of gate maps and the state machines that implement them.

The fifth implication is that the firm must measure these systems like controls, not like models. Standard model metrics (perplexity, generic accuracy) are not the right instruments. The right metrics are operational and governance-oriented. For N1, measure missing-info detection quality, loop exhaustion rates, and escalation clarity. For N2, measure refusal appropriateness, false approvals, and category routing. For N3, measure evidence gap capture, critique quality, and revision convergence. For N4, measure reproducibility, sensitivity to costs, and promotion-to-failure rates in later testing. For N5, measure completion rates under regimes, escalation rates, and tail cost proxies. For N6, measure dissent rates, revision counts, and override rates. For N7, measure inconsistency detection, open-item density, and post-review correction load. For N8, measure retrieval recall on a gold set, citation completeness, and insufficient-evidence rates. For N9, measure alert precision, timeliness, data freshness compliance, and override behavior. For N10, measure vulnerability density, dissent preservation, iteration counts, and whether red-team findings materially change the final

packet. These metrics turn governance from aspiration into practice.

A practical board recommendation, therefore, is to adopt a minimum control standard for any agentic workflow in finance. This minimum standard is not complicated, but it must be mandatory. It includes: explicit typed state; explicit node responsibilities; explicit routing; bounded loops; explicit stop reasons; mandatory artifact export; facts/assumptions/open-items labeling; mandatory human sign-off for any high-stakes endpoint; and change control with regression testing. This standard is what allows speed without fragility. Without it, the organization may move fast but will accumulate hidden liability.

The sixth implication is that change control is not a bureaucratic burden; it is what makes experimentation safe. Agentic workflows are not static. Prompts evolve, policies evolve, and the underlying models evolve. If the institution treats these changes casually, it will not know which version produced which result, and it will not be able to explain variance. This is precisely why `run_manifest.json` and environment fingerprinting are non-negotiable. The institution should define owners for each workflow: a business owner, a technical owner, and a risk/control owner. Any change to routing logic, thresholds, or state schema should require a version bump and a regression run against a fixed test suite. The test suite can be synthetic in early stages, but it must exist. This is how the organization avoids "silent drift" where a seemingly small prompt tweak changes escalation behavior.

The seventh implication is that these architectures provide a pathway to institutional learning that is difficult to achieve with human-only processes. When decisions are made in meetings, the decision rationale is often not structured. When decisions are made with governed workflows, the decision rationale becomes data. Over time, the institution can analyze its own behavior: where does it escalate most frequently, where do open items cluster, what kinds of hypotheses fail later despite early approvals, what kinds of diligence questions frequently hit insufficient evidence, what kinds of treasury events trigger false alarms. This transforms governance from reactive oversight into proactive improvement. It also transforms risk management from anecdotal to empirical.

The eighth implication is cultural. Finance organizations often reward confidence, speed, and persuasion. These incentives are precisely what make ungoverned LLM usage dangerous. The architectures in this handbook are designed to counterbalance those incentives. They reward explicit uncertainty. They reward escalation rather than improvisation. They reward consistency over cleverness. They do not eliminate the need for human judgment, but they discipline how judgment is applied and recorded. For a board, this is a strategic advantage: a firm that can say "we do not know yet, and here is what we need to know" is a firm that avoids the most damaging category of failure, which is acting on false certainty.

What comes next should be framed as an implementation program, not as a research hobby. A practical roadmap has three phases.

In Phase 1 (internal pilot), the workflows run in shadow mode. They produce packets, but humans do not treat them as authoritative. The objective is to validate the mechanics: does the workflow stop correctly, does it escalate correctly, does it produce artifacts consistently, does it preserve facts versus assumptions, does it remain deterministic where it must. The acceptance criteria are governance criteria, not "intelligence"

criteria.

In Phase 2 (controlled adoption), the workflows become standard tools in specific internal processes: intake triage, suitability gating, memo drafting, hypothesis testing, execution planning, rebalancing review, pitch drafting, diligence Q&A, treasury monitoring, research synthesis. In this phase, the workflows are embedded into existing committee and sign-off processes. Humans remain accountable, but they now receive structured packets and audit trails. The acceptance criteria shift to operational metrics: reduction in turnaround time, reduction in inconsistency, improved early-warning timeliness, improved comparability across cases, and measurable reduction in rework.

In Phase 3 (scale and integration), the institution standardizes the workflow patterns as reusable components. This is where LangGraph becomes an institutional platform rather than a set of notebooks. The organization defines shared libraries for state schemas, artifact exports, gate policies, and audit logging. It defines monitoring dashboards and sampling audit procedures. It integrates data governance and access controls. It establishes a formal "workflow registry" where each workflow has owners, versions, and approved use cases. This is the point where the institution can claim it has an agentic capability that is governed, not merely adopted.

A final note on ambition: these architectures are deliberately constrained. They are not designed to maximize autonomy; they are designed to maximize accountability. That is the correct posture for finance. Autonomy can be added later, but only after governance is demonstrated at lower levels of maturity. The board should insist on this sequencing. Capability without controls is not innovation; it is unmanaged risk.

The correct concluding message to executive leadership is therefore straightforward. This handbook proposes a way to use AI that is compatible with professional responsibility. It replaces ad hoc model usage with explicit workflows. It replaces narrative persuasion with structured packets. It replaces hidden assumptions with declared open items. It replaces unlimited iteration with bounded loops and explicit stop reasons. It replaces "trust the model" with "inspect the artifacts." These are the ingredients of an institution-ready approach to agentic systems in finance.

If the organization adopts these principles consistently, it will gain speed where speed is safe, and it will gain discipline where discipline is necessary. It will be able to answer questions faster, but more importantly, it will be able to explain its answers and its decisions under scrutiny. That is what boards should demand, and that is what governance-first agentic architectures are for.

.

References

Bibliography

[1] LangChain, Inc. *LangGraph: Graph API overview* (documentation). LangChain OSS Documentation. Accessed: 19 Feb 2026.

[2] LangChain, Inc. *LangGraph: Use the Graph API* (documentation). LangChain OSS Documentation. Accessed: 19 Feb 2026.

[3] LangChain, Inc. *LangGraph Reference: Graphs* / `StateGraph` (documentation). LangChain Reference Documentation. Accessed: 19 Feb 2026.

[4] Anthropic. *Claude API: API Overview* (documentation). Anthropic Developer Documentation. Accessed: 19 Feb 2026.

[5] Anthropic. *Build with Claude* (developer learning resources). Anthropic Learn. Accessed: 19 Feb 2026.

[6] Board of Governors of the Federal Reserve System. *SR 11-7: Guidance on Model Risk Management*. Supervisory Letter SR 11-7, 4 Apr 2011. Accessed: 19 Feb 2026.

[7] Office of the Comptroller of the Currency (OCC). *Supervisory Guidance on Model Risk Management*. OCC Bulletin 2011-12A, 4 Apr 2011. Accessed: 19 Feb 2026.

[8] National Institute of Standards and Technology (NIST). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1, Jan 2023. Accessed: 19 Feb 2026.

[9] National Institute of Standards and Technology (NIST). *Generative Artificial Intelligence Profile*. NIST AI 600-1, 2024. Accessed: 19 Feb 2026.

[10] Committee of Sponsoring Organizations of the Treadway Commission (COSO). *Internal Control—Integrated Framework: Executive Summary*. May 2013. Accessed: 19 Feb 2026.

[11] Basel Committee on Banking Supervision (BCBS). *Principles for Sound Liquidity Risk Management and Supervision*. Sep 2008. Accessed: 19 Feb 2026.

[12] Basel Committee on Banking Supervision (BCBS). *Basel III: The Liquidity Coverage Ratio and liquidity risk monitoring tools*. BCBS 238, Jan 2013. Accessed: 19 Feb 2026.

[13] Basel Committee on Banking Supervision (BCBS). *Principles for effective risk data aggregation and risk reporting (BCBS 239)*. Jan 2013. Accessed: 19 Feb 2026.

[14] OECD. *G20/OECD Principles of Corporate Governance 2023*. OECD Publishing, 2023. Accessed: 19 Feb 2026.

[15] U.S. Securities and Exchange Commission (SEC). *Regulation Best Interest: The Broker-Dealer Standard of Conduct*. Exchange Act Release No. 34-86031, 5 Jun 2019. Accessed: 19 Feb 2026.

[16] U.S. Securities and Exchange Commission (SEC). *Commission Interpretation Regarding Standard of Conduct for Investment Advisers*. Investment Advisers Act Release No. IA-5248, 5 Jun 2019. Accessed: 19 Feb 2026.

[17] European Securities and Markets Authority (ESMA). *Final Report: MiFID II RTS on order execution policies*. 10 Apr 2025. Accessed: 19 Feb 2026.

[18] European Securities and Markets Authority (ESMA). *ESMA clarification on certain best execution reporting requirements under MiFID II*. 13 Feb 2024. Accessed: 19 Feb 2026.

[19] European Union. *Directive 2014/65/EU (MiFID II)*. 15 May 2014. Accessed: 19 Feb 2026.

[20] FINRA. *FINRA Rule 2210: Communications with the Public*. FINRA Rulebook. Accessed: 19 Feb 2026.

[21] FINRA. *Communications with the Public Rules: Reference Guide*. Publication (Reference Guide), Sep 2020. Accessed: 19 Feb 2026.

[22] CFA Institute. *Standards of Practice Handbook* (12th ed.). CFA Institute, 2024. Accessed: 19 Feb 2026.

[23] International Organization for Standardization (ISO). *ISO 31000:2018 — Risk management — Guidelines*. ISO Standard, 2018. Accessed: 19 Feb 2026.

[24] International Organization for Standardization (ISO). *ISO/IEC 27001:2022 — Information security management systems — Requirements*. ISO/IEC Standard, 2022. Accessed: 19 Feb 2026.

[25] European Union. *Regulation (EU) 2024/1689 (Artificial Intelligence Act)*. 2024. Accessed: 19 Feb 2026.

[26] Almgren, R. and Chriss, N. *Optimal Execution of Portfolio Transactions*. Working paper, 2000.

[27] Kissell, R. *The Science of Algorithmic Trading and Portfolio Management*. Academic Press, 2013.