Agents

Part II: How to Build an Agent

Architectures, Protocols, and Technical Evaluation

Michael J Bommarito II · Daniel Martin Katz · Jillian Bommarito

December 15, 2025

Working Draft Chapter

Version 0.1

This chapter is Part II of a three-part series from the textbook *Artificial Intelligence for Law and Finance*. Part I (What is an Agent?) provides definitions and foundations. Part III (Chapter 08 — Agents Part III: How to Govern an Agent) addresses regulation, risk, and deployment.

The most current copy of the project is available at:

`https://github.com/mjbommar/ai-law-finance-book/`

Contents

1 Introduction

What is an Agent?—Part I of this series (Bommarito et al. 2025)—gave you the GPA+IAT framework for recognizing agentic systems: Goals, Perception, Action, Iteration, Adaptation, and Termination. This chapter answers the next question: *How do you build one?*

Ten architectural questions frame this chapter—questions you should ask when evaluating, deploying, or governing any agentic system. Each corresponds to a capability every useful agent needs: receiving work, understanding requests, finding information, taking action, maintaining context, planning complex work, recognizing completion, escalating when stuck, coordinating with other agents, and operating safely. Behind each capability lie design decisions with real tradeoffs—tradeoffs that determine not just what the system can do, but how reliably it performs, how it fails, and what controls are possible.

The Core Insight

Agents are not magic; they are architecture.
Every capability that makes an agent useful—understanding what you asked, finding relevant information, taking action, remembering context, planning complex work, knowing when to stop, asking for help when stuck—corresponds to a concrete design decision. Those decisions have tradeoffs. Those tradeoffs have consequences.
You do not need to build agents yourself. But you need to know what questions to ask—because architectural decisions made at design time determine what the system can do, how reliably it performs, what can go wrong, and what controls are possible.

1.1 Agents as Professional Teams

The ten questions in this chapter are not arbitrary. They emerge from what agentic systems are designed to do: augment human professionals, automate routine workflows, or eventually replace entire organizational functions. Whatever the ambition, the system must handle the same work those humans and organizations currently perform. A legal research agent must do what a research associate does. A portfolio monitoring agent must do what an analyst does. The capabilities required are not determined by the technology; they are determined by the work. And the work has structure that any system—human or artificial—must accommodate. This yields the design principle that grounds the entire chapter: **an agentic system requires the same structural capabilities as a professional team**.

Consider how a law firm operates. Work arrives through defined channels: client calls, court filings, internal referrals. Associates must understand what partners actually want, not just what they literally said. Research requires access to the right databases with appropriate search strategies. Actions

have consequences—filing a motion, sending a client letter—that require appropriate authorization. Institutional knowledge persists in case files and precedent databases. Complex matters decompose into workstreams with dependencies. Work products have completion criteria. Associates know when to escalate to partners. Teams coordinate across practice groups. And compliance controls ensure the whole operation stays within ethical and regulatory bounds.

A discretionary portfolio management team follows the same pattern. Market data and research flow through defined feeds. Analysts must interpret investment committee mandates. Research requires access to financial databases, company filings, and market intelligence. Trades have real-world consequences requiring compliance checks. Position history and investment theses persist across quarters. Portfolio construction decomposes into sector allocation, security selection, and risk management. Rebalancing has completion criteria. Analysts escalate to portfolio managers when positions approach limits. Teams coordinate across asset classes. And regulatory controls ensure fiduciary compliance.

The structural parallels are not coincidental. Both law firms and investment teams are *cognitive work systems*—organizations that process information, make decisions, and take consequential actions under uncertainty. Agentic systems are also cognitive work systems (Wang et al. 2023; Rao and Georgeff 1995). They face the same architectural challenges and require the same structural capabilities.

This mapping has practical implications. When you evaluate an agentic system, you can ask the same questions you would ask about a professional team. When you design governance for an agentic system, you can draw on the same frameworks that govern professional organizations. When you communicate with technical teams, you can use organizational language they will understand.

1.2 The Ten Questions

These organizational parallels yield ten questions that any agentic system must answer—the same questions you would ask when onboarding a new professional or evaluating a team's capabilities. Table 1 lists those questions in the order an agent encounters them during execution.

Table 1: Architectural questions for agentic systems

Section	Architectural Question
2: Triggers	How does the agent know when it has work to do?
3: Intent	How does the agent understand what is being asked?
4: Perception	How does the agent find things out?
5: Action	How does the agent make things happen?
6: Memory	How does the agent remember things?
7: Planning	How does the agent break a big job into steps?
8: Termination	How does the agent know when it is done?
9: Escalation	How does the agent know when to ask for help?
10: Delegation	How does the agent work with other agents?
11: Governance	How do we keep the agent safe?

Each section addresses one question through organizational analogies, architectural concepts, domain-specific considerations for law and finance, and governance implications. You can read sequentially for cumulative understanding, jump directly to whichever question matters most, or (if you are evaluating vendor claims) skip to the end and start with Section 12. We begin with the first question: how does work reach the agent?

2 How Does an Agent Know When It Has Work to Do?

Consider how work reaches a professional. A client calls with an urgent question, the court docket updates with a new filing, the calendar reminds you that a motion is due tomorrow, and a junior associate realizes an issue exceeds their expertise and brings it to your office. These four channels define how work enters your day: the phone, the inbox, the calendar, and escalation from colleagues.

Agentic systems operate in the same way. A system with tools, memory, and planning capabilities remains idle until work arrives; the architectural question is how tasks enter the system and what events trigger execution.

Triggers

Triggers are the events that start agent execution. In practice, a trigger might be a docket alert, a price crossing a threshold, a calendar deadline coming due, or an internal "I can't proceed safely" signal from the agent itself. Without a trigger, even a highly capable system sits idle.

The distinction between triggers and channels matters for system design. A trigger is the *what*—the event that demands attention. A channel is the *how*—the pathway through which that event reaches the agent. The same trigger (a deadline approaching) might arrive through different channels (a calendar system, an email reminder, a human prompt asking "what's due this week?"). Understanding this separation helps you design systems that can receive work from multiple sources while maintaining consistent processing logic.

For governance, triggers create the audit trail. Every action an agent takes traces back to the trigger that initiated it. When a regulator asks why the system flagged a transaction, or when opposing counsel demands production of the agent's reasoning, you need to show what event started the chain of analysis. Systems that cannot trace actions back to triggers cannot be audited—and in regulated practice, what cannot be audited cannot be deployed.

Channels

Channels are how triggers reach the agent. In professional practice, four channels cover almost all work intake:

External feeds: The world pushes work to you (court filings, market data, regulatory updates).

Human prompts: People request work directly (chat, email, collaboration platforms).

Scheduled jobs: Time itself triggers execution (deadlines, periodic checks, end-of-day).

Escalation events: Internal signals that ask for human help (budget exhaustion, low confidence).

The four channel types serve different operational needs. External feeds enable reactive monitoring, allowing the system to respond to events as they occur in the world. Human prompts enable interactive collaboration, letting the system work alongside professionals who direct its attention. Scheduled jobs enable proactive workflows that ensure routine tasks happen without requiring someone to remember to initiate them. And escalation events close the loop on human oversight by ensuring the system asks for help when it reaches its limits, a topic we explore in depth in Section 9.

Before an agentic system can reason or act, it must first notice that work exists. Channels are the sensory apparatus of the system: the ways it becomes aware of its environment and the tasks it must accomplish—just as a lawyer cannot respond to a motion they never received.

2.1 External Feeds: The World Pushes Work to You

External feeds deliver events from systems outside the agentic system's direct control. The external system pushes notifications when events occur, much like receiving service of process rather than checking the courthouse daily to see if you have been sued.

Legal and Regulatory Feeds: Court docket systems like CM/ECF and state e-filing platforms send

Figure 1: Four channel types through which work reaches agentic systems. External feeds push events from outside systems; human prompts arrive through interactive interfaces; scheduled jobs trigger on time-based conditions; and escalation events signal internal limits requiring human intervention. All channels converge on the agentic system's event router.

notifications whenever documents are filed in cases you are monitoring (Administrative Office of the U.S. Courts 2024b; Administrative Office of the U.S. Courts 2024a). When an alert arrives, an agentic system can retrieve the filed document through PACER, analyze its contents, and trigger the appropriate response—whether that means flagging a motion for attorney review or updating a case timeline. The SEC's EDGAR system offers similar capabilities for corporate filings, with RESTful APIs providing real-time access to submissions with sub-second processing delays (U.S. Securities and Exchange Commission 2024), allowing agentic systems to monitor competitors' 10-Ks and flag material differences from your company's disclosures. Regulatory agencies publish updates through the Federal Register and agency websites, while citator services like Westlaw and Lexis can notify the system whenever monitored cases are cited or overruled.

Financial Market Feeds: Financial institutions receive real-time market data through providers like Bloomberg and Reuters. A portfolio management agentic system can subscribe to price alerts and receive notifications when thresholds are crossed, then evaluate rebalancing rules and either execute trades within risk limits or escalate to a portfolio manager for approval. These events cascade

through financial systems as trades trigger position updates, which in turn trigger risk recalculation, compliance checks, and dashboard refreshes. News feeds add another layer by delivering headlines, earnings announcements, and sentiment analytics, allowing agentic systems to assess materiality and alert managers when developments appear significant.

Speed vs. Reasoning: A Critical Distinction

Market data arrives at millisecond granularity. LLM-based reasoning operates at second-to-minute timescales. This fundamental mismatch determines where agentic systems add value in financial workflows.

Not suited for agentic systems:

- High-frequency trading and market-making
- Latency-sensitive execution
- Any task requiring microsecond response times

Well suited for agentic systems:

- Strategic portfolio decisions and rebalancing analysis
- Investment thesis development and research synthesis
- Compliance monitoring

The architecture pattern: Fast deterministic systems handle real-time data capture and threshold detection. When a threshold triggers—a position approaching its limit, a price target hit, an anomaly detected—it generates an event that the agentic system processes. Keep the microsecond path deterministic; hand off to the agentic system only once an alert is raised.

Integration Patterns: External feeds reach agentic systems through **webhooks** (HTTP callbacks pushing events from external systems) or **message queues** (durable event streams such as Kafka or RabbitMQ that provide ordered delivery and replay) (Free Law Project 2023). Webhooks work well for low-volume, time-sensitive events where immediate delivery matters and occasional missed events are acceptable. Message queues provide ordering, durability, and replay capabilities essential for regulated applications requiring audit trails (Using Event Sourcing and CQRS to Build a High Performance Point Trading System 2019). In practice, many systems use both: a portfolio management system might use webhooks to receive immediate notification when a stock price crosses a stop-loss threshold, while using message queues to process daily trade confirmations that require guaranteed delivery and audit logging. Research on event-driven architectures demonstrates measurable performance benefits over monolithic approaches for such event-processing workflows (On the Impact of Event-Driven Architecture on Performance: An Exploratory Study 2023).

2.2 Human Prompts as Events

Human prompts feel different from external feeds because they are interactive and synchronous. You type something and expect a response. But at the architectural level, a human prompt is just another event type. The user generates an event, the system receives it through a channel, processes it, and responds. Treating prompts this way actually simplifies design, because all events can flow through the same routing and prioritization logic rather than requiring separate code paths for "chat" versus "background" work.

Chat interfaces offer the most direct channel for human interaction. An associate might type "Find Fifth Circuit authority on personal jurisdiction for e-commerce defendants," and the system searches relevant databases, presents summaries, and waits for follow-up refinements. An analyst asks for revenue growth comparisons across portfolio companies, receives a table, and requests additional filtering. What makes chat powerful is its support for iterative clarification. Each message is simply another event processed through the standard loop, just with tighter latency expectations than background tasks.

Synchronous interfaces like chat create design constraints that asynchronous channels do not. When a user is waiting for a response, every second of silence feels like something has gone wrong. Systems designed for direct interaction need to prioritize lower latency and provide transparent, regular feedback about what is happening. This might mean streaming partial responses as they are generated, displaying progress indicators that show which tools the system is invoking, or breaking complex tasks into visible steps so users can see the work unfolding. Without this feedback, a blank screen followed by a complete response thirty seconds later leaves users uncertain whether the system is working, stuck, or has crashed. Good conversational design treats feedback as a feature, not an afterthought. Acknowledge the request immediately, show progress throughout, and deliver results incrementally when possible.

Email routing lets agentic systems process work that arrives through existing communication channels. A general counsel might forward a business unit's compliance question to a monitored mailbox, and the system extracts the question, searches relevant guidance, and emails back an assessment. The main challenge here is intent classification, since email bodies tend to be unstructured and often include forwarded threads with multiple topics buried in the conversation.

Collaboration platforms like Slack and Teams allow agentic systems to appear as team members in the workflow. Users can @mention the system in channels, send direct messages, or invoke capabilities through **slash commands** (e.g., `/research "personal jurisdiction"`). A litigation team discussing strategy can request research directly in their coordination channel without switching applications. Security becomes important here because collaboration platforms typically log all responses, and channels may include viewers who should not have access to certain information.

Voice interfaces work best for short, urgent requests where typing is impractical. Think of a portfolio manager checking a position while walking between meetings, or a lawyer needing a quick

case citation during a call. The tradeoffs are real. Transcription errors can mangle legal jargon, turning "Chevron deference" into something unrecognizable, and authentication is harder without a keyboard. For high-stakes requests, voice interfaces should require explicit confirmation before the system takes action.

In contrast, asynchronous channels like email and background automation do not require the same real-time feedback, but they introduce different design challenges. When users are not watching, systems still need observability through logs, dashboards, and notifications that let operators understand what the system did and why. And because asynchronous requests lack the back-and-forth of conversation, getting intent right on the first pass becomes critical. A user who submits a request and walks away cannot clarify a misunderstood instruction until they see the wrong output hours later. This is why intent understanding (Section 3) and perception design (Section 4) matter even more for asynchronous workflows. The system must extract what the user actually wants from a single message, gather the right information without guidance, and produce results that match expectations without iterative correction.

2.3 Scheduled Jobs: Time as Trigger

Some work follows predictable schedules rather than arriving from external events or human prompts. End-of-day reconciliation, monthly compliance reporting, quarterly reviews, and annual filings all happen on a calendar rather than in response to some triggering event. For these recurring tasks, time itself becomes the trigger.

Calendar-driven deadlines govern much of legal practice. You must answer the complaint within 21 days, file motions 30 days before hearings, and respond to discovery within 30 days. Agentic systems can monitor litigation calendars, calculate deadlines while accounting for court holidays, schedule reminders as deadlines approach, and escalate if work remains incomplete. More sophisticated deadline systems go further by retrieving the complaint, extracting claims, generating draft answers with standard defenses, and presenting those drafts for attorney review before filing. Financial institutions face similar deadline-driven work that spans SEC reporting deadlines, tax filings, and contractual obligations to lenders.

Periodic compliance checks run even when no external event triggers a review. An investment compliance system might run nightly to check portfolios against client guidelines and flag any violations it finds. A law firm conflicts system retrieves new docket entries each day, extracts party names, and checks them against the conflicts database. These scheduled checks enable continuous monitoring that would be impractical to perform manually across thousands of matters or client accounts.

End-of-day workflows are particularly important in financial institutions, where teams need to reconcile trades, calculate valuations at market close, generate P&L reports, and prepare risk reports for the next morning. When the market closes, an EOD system retrieves final prices, marks positions

to market, calculates P&L, and identifies any unexplained variances. The system then distributes reports to stakeholders, and if any step fails, it escalates rather than proceeding with incomplete data. Law firms run similar periodic workflows that remind attorneys to enter time, generate draft invoices at month-end, and flag anomalies for partner review.

2.4 Escalation Events: When Agents Reach Their Limits

The previous three channel types bring work into the agentic system from outside, but escalation events operate internally. When an agentic system reaches a limit and cannot proceed autonomously, it generates an event signaling that it requires human intervention. This transfers control to human decision-makers at precisely the moments when human judgment matters most.

Four escalation triggers appear most frequently. **Budget exhaustion** occurs when the system approaches resource limits such as token consumption, iteration counts, time limits, or cost caps, and must decide whether to stop or request additional budget. **Low confidence** triggers escalation when uncertainty is too high for autonomous action, whether due to conflicting authority, novel situations, or results that seem implausible. **Approval requirements** force escalation for certain actions that require explicit human authorization regardless of the system's confidence, such as filing court documents, sending client communications, or executing large trades. Finally, **errors and anomalies** trigger escalation when tools fail repeatedly, data is inconsistent, or the system detects red flags that require human investigation. Section 9 addresses when and how agentic systems should escalate to humans in greater detail.

2.5 Surfaces: How Users Experience Agentic Systems

A **surface** is the interaction modality through which users encounter an agentic system. The same underlying capabilities, including intent understanding, tool use, memory, and planning, can manifest through radically different user experiences. Usability researcher Jakob Nielsen argues that generative AI represents the first new user interface paradigm in sixty years, marking a shift from *command-based interaction* where you tell the computer what to do, to *intent-based outcome specification* where you tell the computer what you want (Nielsen 2023). But intent-based systems still require interfaces, and those interfaces shape how effectively users can express intent and consume results.

Interaction Surfaces

An **interaction surface** is the user-facing modality through which humans engage with an agentic system. The same underlying capabilities can manifest through radically different user experiences depending on four key dimensions: synchronicity, initiative, embodiment, and output format.

Synchronicity determines when results arrive. Synchronous interactions deliver results while the

user waits, as with chat interfaces where you ask a question and watch the response stream in. This suits exploratory work where you refine direction based on what you see. Asynchronous interactions deliver results later through notifications or reports, as with document generation where you specify requirements, do other work, and return when the draft is ready. This suits production tasks where waiting would waste billable time.

Initiative captures who starts the conversation. In pull models, the system waits for the user to initiate. A research assistant that answers questions when asked operates this way, giving the user control over when and whether to engage. In push models, the system reaches out when something needs attention. A docket monitor that alerts you to new filings operates this way, deciding when to interrupt based on relevance and urgency.

Embodiment refers to whether the system is visible in the workflow. Visible systems appear as explicit participants, like a chat avatar or copilot sidebar that makes the agentic system's presence clear. Users know they are interacting with AI and can direct it consciously. Invisible systems operate as infrastructure, like a compliance screening system that flags suspicious transactions in the background. Users experience the outputs without seeing the system that produced them.

Output format determines what the system produces. Conversation turns suit iterative exploration where direction emerges through dialogue, producing chat transcripts from research queries, brainstorming, and strategy discussions. Structured documents suit formal deliverables with defined formats, producing research memos, due diligence reports, and contract summaries ready for distribution. Actions in the world suit automation workflows, where filing a document, sending an alert, or executing a trade produces effects rather than text.

Three primary surfaces dominate current deployments, each suited to different task types and user contexts.

Conversational surfaces present the agentic system as an interactive dialogue partner. The user types or speaks, the system responds, and the user refines based on what they see. This is the paradigm of ChatGPT, Claude, and embedded copilots. Conversational surfaces excel at *exploratory tasks* where users refine direction through iteration, such as a partner thinking through case strategy, an analyst exploring market scenarios, or an associate researching an unfamiliar area of law. The interaction is synchronous and user-initiated.

Conversational surfaces have limitations, however. They require users to articulate intent in natural language, which Nielsen calls the "articulation barrier." They lack the **affordances** (design cues suggesting possible interactions) of graphical interfaces, offering no menus to browse and no buttons to discover capabilities. And they demand attention, since the user must remain engaged throughout the interaction.

Automation surfaces present the agentic system as invisible infrastructure that monitors, analyzes, and acts in the background. Users receive outputs only when something relevant happens. Examples

include portfolio surveillance that alerts when positions breach limits, docket monitoring that flags new filings in active matters, and compliance systems that screen transactions against sanctions lists. The interaction is asynchronous and system-initiated.

Automation surfaces suit *monitoring tasks* where continuous human attention is impractical. A compliance officer cannot manually review every transaction, and a litigator cannot check every docket daily. The agentic system handles the routine cases, surfacing only exceptions that require human judgment. The user experience is defined by what *does not* happen, since no alert means no problem.

Document surfaces present the agentic system as a drafting assistant that produces structured work products such as research memos, due diligence reports, contract summaries, and client presentations. The user specifies requirements, the system produces a document, and the user reviews, edits, and distributes. The interaction is asynchronous because the system works while the user does other things, but it remains user-initiated.

Document surfaces suit *production tasks* with defined deliverables. The associate needs a memo for the partner's review, and the analyst needs a report for the investment committee. The output format matters here because you need a polished document that can be filed, sent to clients, or presented to regulators, not a chat transcript.

Matching Surface to Task

Surface selection is a design decision, not a technical constraint. When choosing how users will interact with an agentic system, match the surface to the task type.

Exploratory tasks like research questions, strategy discussions, and ad hoc analysis work best through **chat surfaces** that support iterative refinement.

Monitoring tasks like docket surveillance, compliance screening, and portfolio alerts work best through **automation surfaces** that operate in the background.

Production tasks like research memos, due diligence reports, and regulatory filings work best through **document surfaces** that generate formal deliverables.

Many deployments combine all three. A litigation support system might offer chat for research, automation for alerts, and document generation for motion drafts. The underlying reasoning capabilities remain constant while only the interaction modality changes.

Mismatches waste effort. Forcing chat onto monitoring tasks demands attention no one can sustain. Forcing documents onto exploratory tasks prevents the iteration that produces good results.

Emerging surfaces extend beyond these three patterns. *Embedded copilots* integrate agentic capabilities directly into existing applications like Word, Excel, or domain-specific software, combining

familiar graphical interfaces with intent-based interaction. *Ambient interfaces* use voice or environmental sensors to enable hands-free interaction, though transcription errors and authentication challenges limit their use in high-stakes applications. *Agentic APIs* expose capabilities to other software systems rather than human users, enabling machine-to-machine orchestration.

2.6 Evaluating Trigger Systems

When evaluating agentic systems, whether you are building or buying, you should assess trigger capabilities against five criteria.

Coverage asks whether the system receives events from all relevant sources. A litigation system that monitors CM/ECF but not state court dockets has incomplete coverage that could miss critical filings. **Latency** measures how quickly events reach the system. Real-time market data requires sub-second delivery, while docket alerts can tolerate delays of several minutes. **Reliability** addresses what happens when feeds fail. Robust systems need retry logic, fallback sources, and alerting when data goes stale. **Priority mechanisms** determine whether the system can distinguish urgent events from routine ones. During a market crash or litigation crisis, the right events must reach the right handlers immediately. Finally, **auditability** ensures that every trigger is logged. When a regulator asks why the system took action, you need a complete record of the triggering event. The following chapter on governance addresses audit requirements, regulatory compliance frameworks, and governance controls in detail.

2.7 From Triggers to Action

Triggers answer how work reaches the agentic system, but triggering is only the beginning. Once an event arrives, the agentic system must:

- **Understand intent**: What is being asked?
- **Perceive information**: What does the system need to know?
- **Take action**: What should the system do?
- **Remember context**: What should persist across sessions?
- **Plan execution**: How should work be decomposed?
- **Recognize completion**: When is the task done?
- **Escalate when needed**: When should humans intervene?

The connection between questions is direct. An external feed delivers a court filing notification. The router classifies it as urgent litigation work and dispatches to the litigation agentic system. The system retrieves case context from memory, downloads the filed document through PACER, analyzes content, searches for responsive authority, generates deadline calculations, and drafts a response strategy. At each step, the system might escalate: low confidence in legal analysis triggers escalation to a senior litigator; filing a responsive document requires approval; approaching budget limits prompts a status update.

Section 3 examines the next question: once work arrives, how does the agentic system understand what's being asked?

3 How Does an Agent Understand What's Being Asked?

When a partner walks into your office and says "look into the Johnson matter," your first job is understanding what that actually means. You must determine whether this is a quick status check or a request for deep analysis, whether you should answer a specific question or identify all issues, and whether this is urgent work for today's call or background work for next week's meeting. The words you hear are the **instruction**; the underlying purpose that those words point toward is the **intent**.

Every professional develops this skill over time: reading the assignment memo, clarifying ambiguous instructions, and understanding not just what was said but what was meant. Junior associates tend to over-clarify; senior associates internalize firm norms and client expectations and infer appropriately. The best professionals know when to ask and when to proceed.

Agentic systems face the same challenge. The user provides an instruction: natural language, often ambiguous, sometimes contradictory. The agent must extract intent: what goal is being pursued, what constraints apply, what success looks like. This is the second fundamental question: *How does an agent understand what's being asked?*

Four concepts structure this process: instruction, intent, goal, and task. Understanding how they relate—and where gaps arise—is essential for building and governing agentic systems.

Instruction

An **instruction** is the words the user provides—the raw input that starts the process. "Review this credit agreement for risks." "Rebalance to reduce tech exposure." "Look into the Johnson matter."

Instructions are where work begins, but they are rarely complete specifications. The word "risks" raises immediate questions: risks to whom? The lender or borrower? Material risks or all risks? Legal risks, financial risks, or both? "Reduce tech exposure" leaves open the target level, the mechanism (sales, hedges, or both), and tax and timing constraints. "Look into" specifies neither depth, urgency, nor deliverable format. Instructions are clear enough to start; they are not clear enough to finish.

Pre-LLM systems handled instructions through rigid steps: matching exact words, filling forms, simple if-then checklists. These systems worked for narrow domains with controlled vocabularies but broke on natural language variation. "Find cases on personal jurisdiction" and "What's the law on where you can sue someone?" express similar meanings but look nothing alike to a keyword matcher.

Intent

Intent is the underlying purpose, constraints, and success criteria behind an instruction. Intent captures the human meaning—what the user actually wants, not just what they said. Where an instruction might be "review this credit agreement," the intent might be "identify material risks to the lender by tomorrow for the partner's client call."

The gap between instruction and intent has always existed; what has changed is our ability to bridge it. Large language models dramatically improved intent inference from natural language. Where rule-based systems required exact matches, LLMs handle variation, implicit context, and domain-specific jargon. Modern LLMs excel at handling noisy input (misspellings, shorthand, tangential information), resolving references using conversational context, inferring domain-specific meaning, and detecting implicit constraints that professionals take for granted.

Despite these capabilities, intent understanding remains imperfect. As conversations extend, LLMs may lose track of earlier context or constraints. When clarification is needed, LLMs sometimes proceed with a default interpretation rather than asking, resulting in a "helpful but wrong" failure: the agent does *something* reasonable rather than confirming it understood correctly. Professionals also communicate through implication. Phrases like "This needs to be right" signal high stakes, while "When you get a chance" signals low urgency. These signals may not be explicitly parsed by current models.

Goal

A **goal** is the machine-readable specification that intent points toward—the structured outcome that would satisfy the request. Goals connect to the GPA+IAT framework introduced in the previous chapter: an agent pursues goals through perception and action. Where intent is "identify material risks to the lender by tomorrow," the goal is "produce a lender-risk memo that meets firm policy and is ready for partner review."

Goals provide the target for planning and the criterion for termination. An agent that understands the goal can determine whether its work is complete: does this memo identify the material risks? Does it meet firm standards? Is it ready for the partner? Without a clear goal, the agent cannot know when to stop—the "runaway associate" problem of endless research without a deliverable.

Task

A **task** is the concrete unit of work the agent executes to advance a goal. A single goal typically decomposes into multiple tasks. For the lender-risk memo, tasks might include: extract financial covenants, compare covenant terms to market standards, identify collateral coverage gaps, flag cross-default provisions, and draft the summary memo.

Tasks are where planning meets execution. Section 7 addresses how agents decompose goals into task sequences; Section 8 addresses how agents recognize when tasks—and goals—are complete.

The flow from instruction to task is: **Instruction → Intent → Goal → Tasks**. Intent bridges the gap between what users say and what they mean. Goals give agents targets to aim for. Tasks give agents concrete work to execute. Failures at any stage propagate forward: misunderstood instructions yield wrong intent; wrong intent yields wrong goals; wrong goals yield wasted tasks.

3.1 Bridging the Gap

Traditional enterprise systems classified work using explicit **routing rules**: fixed logic that checked message details and forwarded them to the right handler. A court filing tagged `matter_id=12345` goes straight to litigation monitoring; a portfolio company's SEC filing routes to compliance review. This pattern dominated middleware architecture for decades: message queues, **enterprise service buses (ESBs)**, and workflow engines functioned like a mailroom sorting notices to the right department.

LLM-based agentic systems replace explicit routing logic with semantic reasoning. Rather than maintaining explicit rules for every event type, the model *understands* what kind of work this is. "Review this credit agreement" and "Check this loan doc for problems" express similar intents despite different words; the LLM recognizes both as document review tasks without explicit rules mapping each phrase to a handler.

This architectural shift creates two primary tensions in system design.

The first tension is between flexibility and predictability. Rule-based routing is deterministic: the same input always produces the same classification. LLM-based classification relies on likelihoods and may shift slightly with model updates, prompt phrasing, or context. For regulated applications, this requires additional governance through logging classifications, monitoring for drift, and maintaining override rules for critical categories.

The second tension involves explicit versus implicit knowledge. Routing rules encode domain knowledge explicitly in code, requiring developers to anticipate every category and write rules to match. LLM classification absorbs domain knowledge implicitly through training, recognizing that legal research and case analysis are related without explicit rules. But implicit knowledge is harder to audit because there is no specific rule to inspect, and it may reflect training biases.

Production systems often combine both patterns. Simple, high-volume, time-sensitive classifications use deterministic rules, so a margin call always routes to the trading desk. Complex, ambiguous, or novel requests use LLM reasoning. The rule-based layer handles predictable cases efficiently while the LLM layer handles everything else.

For architects evaluating agentic systems: where must classification be deterministic (regulatory requirements, latency/speed constraints, auditability)? Where does flexibility justify the overhead of

LLM reasoning? The answer shapes system design.

Intent Inference Is Not Mind Reading

LLMs infer *probable* intent from language patterns; they do not read minds. Inference fails when:

- The instruction is genuinely ambiguous (multiple reasonable interpretations)
- The user's intent differs from typical patterns for similar language
- Critical context exists outside the conversation (prior meetings, firm norms)
- The user themselves is unclear about what they want

Design for clarification, not guessing. The aim is an agent that surfaces uncertainty and asks, not one that pretends certainty.

3.2 Goal Extraction from Natural Language

Once the agent receives an instruction, it must extract structured goals that can guide execution. This extraction transforms natural language into actionable specifications through two main processes: intent classification and constraint recognition.

The first step, intent classification, translates the instruction into task types that determine workflow. This step converts raw words into candidate tasks that can advance the underlying goal:

- **Information retrieval**: "What's the current NAV (Net Asset Value)?" "Find the latest 10-K"
- **Research and analysis**: "Research whether we can pierce the corporate veil (hold shareholders liable)"
- **Document review**: "Review the acquisition agreement for change-of-control provisions"
- **Document generation**: "Draft an engagement letter for the Smith matter"
- **Calculation**: "Calculate the IRR (Internal Rate of Return) assuming a 5-year hold"
- **Monitoring**: "Alert me if tech exposure exceeds 30%"

Different task types invoke different tools, planning patterns, and success criteria. A research task requires search and synthesis; a calculation task requires structured computation; a monitoring task requires continuous observation.

Beyond classification, the agent must also recognize entities and constraints. Entity recognition identifies what the task concerns: matters, clients, securities, parties, documents, and jurisdictions. When someone says "Review the Smith acquisition agreement," the agent must recognize a reference to a specific document; "Research Delaware fiduciary duties" references a jurisdiction that shapes which law applies.

Constraint recognition identifies what bounds apply to execution. Temporal constraints include deadlines, as-of dates, and time windows. "By Friday" sets a deadline; "as of year-end 2024" sets a reference date; "over the past quarter" defines a window for analysis. Resource constraints set

budget and effort limits, with phrases like "Spend no more than 2 hours" or "focus on Articles 3 and 4" bounding scope. Format constraints specify how deliverables should appear: "Summarize in one page" constrains length, "prepare a memo for the file" specifies format, and "I need something to show the client" signals an external audience requiring different tone and detail.

Two additional constraint types often remain unstated. Audience and privilege constraints determine who will see the output and what confidentiality must be preserved. Risk and compliance constraints set limits that professionals internalize but rarely articulate: a compliance review implicitly requires flagging violations, and a client communication implicitly requires privilege protection.

Once extracted, these components can be organized into structured goal representations that guide execution. These representations resemble short assignment memos that the planning system can act on (Section 7) and the termination system can measure against (Section 8).

Structured Goal Representation: Document Review

Task type	Document review
Document	Smith Acquisition Agreement
Objective	Identify change-of-control provisions
Constraints	
Deadline	January 15, 2025
Scope	Sections 5–8
Deliverable	Summary memo
Success criteria	
•	All CoC provisions identified
•	Triggering events listed
•	Consent requirements noted

3.3 Ambiguity Detection and Clarification

Not all instructions can be unambiguously interpreted. The agent must detect ambiguity and decide whether to clarify or proceed. The decision depends on two factors: ambiguity severity and action stakes.

When both stakes and ambiguity are low, the agent should proceed with its best interpretation. If someone asks "What's Apple's market cap?" and there is slight uncertainty about whether they mean Apple Inc. or Apple Hospitality REIT, the dominant interpretation is obvious and the cost of being wrong is low since correction is easy. When stakes remain low but ambiguity is high, a brief clarification prevents wasted effort. If someone asks "Research the statute of limitations" without

specifying the claim type, a quick question saves hours of potentially misdirected work.

When stakes are high but ambiguity is low, the agent should confirm before acting. If the instruction is clear but consequential, such as "File this motion," confirmation prevents irreversible errors even when the agent is confident it understood correctly. When both stakes and ambiguity are high, thorough clarification is essential. If someone says "Handle the regulatory response" for a complex matter, extended clarification is appropriate before taking any action.

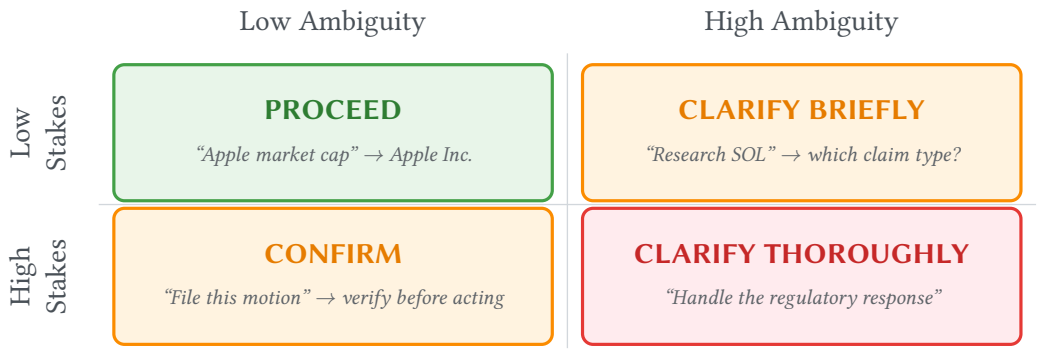

Figure 2: Decision matrix for when agents should clarify user intent. Stakes measure consequence of error; ambiguity measures interpretation confidence.

Effective clarification has four characteristics. It is specific, asking "Which jurisdiction's statute of limitations: Delaware or New York?" rather than "Can you clarify?" It is contextual, referencing what the agent already understands: "I understand you want me to review the credit agreement. Should I focus on lender protections, borrower obligations, or both?" It is actionable, offering options rather than open-ended questions: "Should I (a) provide a comprehensive review of all provisions, (b) focus on the financial covenants, or (c) flag only provisions that differ from our standard template?" And it is bounded, limiting clarification rounds. If the agent needs extensive clarification, it may be the wrong tool for the task, or the user may need to think through requirements before delegating.

> *Poor clarification*: "Can you clarify?"
> *Better*: "Should I assess lender risks, borrower risks, or both?"
> *Best*: "You asked to reduce tech exposure. Should I (a) sell tech to 25% target, (b) hedge with options, or (c) add non-tech positions? Which deadline matters—this week or month-end reporting?"

Research has documented that LLMs sometimes select a default interpretation rather than asking for clarification, even when ambiguity is significant (Zhang et al. 2024; Wang et al. 2024a). This "proceed without asking" behavior creates real risk: the agent interprets "review the contract" as a surface-level summary when the user expected deep issue-spotting, delivering work product that is technically responsive but wrong.

Several strategies can mitigate this tendency: prompt engineering that emphasizes clarification for

ambiguous requests, confidence thresholds that trigger clarification below a certainty level, user training to provide detailed initial instructions, and checkpoint reviews before significant work begins. From a governance perspective, teams should monitor for cases where the agent proceeded confidently but delivered unexpected results. These cases may indicate calibration problems in ambiguity detection.

3.4 Constraint Identification

Beyond explicit instructions, agents must identify constraints that bound acceptable execution. These constraints fall into several categories that often interact.

Temporal constraints include deadlines and time windows. Some are explicit, like "by Friday." Others are implicit, such as court filing deadlines calculated from procedural rules. Still others are contextual: "before the board meeting" requires knowing when the meeting is scheduled. Resource constraints set budget and effort limits. Token budgets limit API costs; time budgets limit calendar impact; scope constraints focus effort on high-value areas.

Scope constraints define what is in and out of bounds. "Focus on Articles 3 and 4" excludes other articles; "just the Delaware analysis" excludes other jurisdictions. Format and style constraints specify how deliverables should appear: memo versus email versus presentation, formal versus casual tone, internal versus client-facing audience. Risk and compliance constraints specify what must be avoided: privilege protection, conflicts of interest, regulatory restrictions, and confidentiality obligations. These constraints often apply implicitly based on context.

Professionals operate under many constraints they rarely state explicitly. When a partner says "research Section 10(b) liability," implicit constraints include:

- Use authoritative sources (binding precedent, not blog posts)
- Focus on the relevant jurisdiction (probably the circuit where the case is filed)
- Assume current law (not historical analysis unless specified)
- Protect privilege (don't disclose strategy in external searches)
- Operate within budget norms (don't spend 40 hours on a 2-hour task)

Agents must infer these constraints from context, domain knowledge, and organizational norms. Memory systems (Section 6) help by preserving firm-specific expectations; user profiles track individual preferences; matter context provides case-specific constraints.

3.5 Validation and Domain Examples

Before executing, agents should validate their understanding of intent. Several patterns support this validation.

reflection and summarization involve the agent pausing—much like an associate double-checking their notes—to restate its understanding before proceeding, giving the user an opportunity to correct

misunderstandings before work begins. **chunked validation**, like partner check-ins after each memo section, breaks complex tasks into phases rather than validating all at once. After completing research, the agent summarizes findings and confirms direction before drafting. After drafting, it confirms the approach before finalizing. Each checkpoint prevents error propagation.

Confidence signaling requires the agent to indicate how confident it is in its own understanding. When confidence is high, the agent can proceed with light oversight; when confidence is low, the right move is to pause and ask for clarification rather than press ahead. Clear confidence signaling helps users decide how much review is needed and whether to treat the output as a draft, a starting point, or a near-final product.

Consider how intent extraction and validation work together for a legal task. Given the instruction "Review this credit agreement for risks," the agent classifies this as a document review task and detects that "risks" is ambiguous (risks to whom? what types?). Context gathering reveals this is a lender-side engagement for a senior secured facility. The agent infers implicit constraints (focus on lender risks, prioritize material issues, assume current market terms as baseline) and clarifies: "I'll review from the lender's perspective, focusing on credit risk, collateral coverage, and covenant adequacy. Should I also flag documentation risks (drafting issues, missing provisions) or focus only on substantive credit terms?"

Extracted Goal: Credit Agreement Review

Task type	Document review
Perspective	Lender
Document	Attached credit agreement
Focus areas	Credit risk, collateral, covenants
Deliverable	Risk summary memo

Success criteria

- Material risks identified
- Comparison to market terms
- Recommendations provided

The agent validates: "I'll prepare a memo identifying material risks to the lender, comparing key terms to market standards, and recommending negotiation points. I'll have a draft for your review by tomorrow afternoon."

The same pattern applies to financial tasks. Given the instruction "Rebalance to reduce tech exposure," the agent classifies this as a portfolio action task and immediately detects multiple ambiguities: how much reduction? through what mechanism? with what constraints? Context gathering

reveals current tech exposure at 35% against a 25% target. Through clarification dialogue, the agent confirms the user wants to reach target through sales while minimizing tax impact (preferring loss harvesting—selling assets at a loss to offset gains—and long-term gains over short-term).

Extracted Goal: Portfolio Rebalancing

Task type	Portfolio rebalancing
Objective	Reduce tech exposure
Target	25%
Mechanism	Sales
Constraints	
•	Tax optimization enabled
•	Prefer loss harvesting (offset gains)
Approval required	Yes

The agent validates: "I'll prepare a rebalancing plan to reduce tech from 35% to 25% through sales, optimized for tax efficiency. I'll present the plan for your approval before executing any trades."

Intent Understanding Is Continuous

Intent extraction is not a one-time step at task initiation. As the agent works, it may discover:

- The original understanding was incomplete (new constraints emerge)
- The user's intent has evolved (priorities shift mid-task)
- Implicit constraints conflict (cannot optimize for both)
- The task is impossible as specified (constraints are mutually exclusive)

Effective agents surface these discoveries through clarification rather than proceeding with outdated or impossible goals. Intent understanding is iterative, not instantaneous.

Intent understanding connects to other framework questions. Memory (Section 6) improves intent extraction over time by preserving user preferences, matter history, and firm norms. Planning (Section 7) depends on clear intent; extracted goals feed the planning system, while ambiguous intent propagates through the plan as uncertainty. Governance must address intent misalignment as a core risk, verifying goal alignment before deployment and monitoring for drift during operation.

Understanding intent bridges the gap between what users say (instruction) and what they mean (intent), shaping the goals and tasks the agent will plan. Clarification beats guessing when ambiguity is significant and stakes are high. Constraints—time, scope, audience, compliance, and budget—matter as much as goals. Validation prevents wasted effort by confirming understanding before significant work begins.

With triggers delivering work and intent extraction revealing what's being asked, the agent faces a practical problem: extracted goals require information the agent does not yet have. The credit agreement analysis task requires the actual credit agreement. The research question requires access to case law databases. The rebalancing plan requires current portfolio positions and market prices. Understanding what you need to do is not the same as having what you need to do it.

Section 4 examines the next question: how does an agent find things out? Perception tools—the interfaces to external information sources—bridge the gap between understanding a task and executing it.

4 How Does an Agent Find Things Out?

The previous section examined how agents understand instructions—extracting goals, detecting ambiguity, and gathering context. However, understanding a task differs from completing it. An agent that correctly interprets "Research Ninth Circuit authority on personal jurisdiction" still requires access to case law databases. An agent that processes the instruction "Analyze this credit agreement from the lender's perspective" still requires the credit agreement itself.

Human professionals face similar constraints. A junior associate's effectiveness depends on access as much as reasoning. The ability to query Westlaw, access Bloomberg terminals, and search the firm's precedent database determines which problems the associate can solve. Without access to information sources, even capable professionals reason in a vacuum.

Agents face the same constraint. An LLM can reason about legal and financial concepts, but without *perception tools*—interfaces to external information—it cannot access current case law, market prices, client documents, or regulatory filings. Perception tools are the interface mechanism through which agents observe the world.

Tools and Perception

A **tool** is a function that allows an agent to interact with external systems. **Perception tools** are read-only: they observe without changing the world. The agent queries a database, retrieves a document, or fetches market data, while the external system's state remains unchanged.

Perception implements the "P" in the GPA+IAT framework. It defines the boundary of what information the agent can access to inform its reasoning.

In this section, we examine perception: the read-only tools that enable agents to gather information. Section 5 then examines action: the write tools that enable agents to effect change. This distinction is critical for governance, as read operations carry different risks than write operations.

4.1 Perception Tool Categories

Effective perception requires selecting the correct tool for the information type. Perception tools generally fall into three categories: information retrieval, document processing, and computation.

Information retrieval tools query authoritative platforms. Legal research tools connect to services like Westlaw, Lexis, or PACER (Public Access to Court Electronic Records) (Administrative Office of the U.S. Courts 2024b). These tools allow agents to search case law, retrieve opinions, and download federal court filings. Financial research tools connect to platforms like Bloomberg, FactSet, or Refinitiv. These enable agents to query real-time prices, retrieve fundamentals, and access analyst research. Internal knowledge bases include document management systems (DMS) and deal archives. Accessing these allows the agent to retrieve prior work product and precedent.

Document processing tools transform raw files into structured data. Text extraction converts PDFs and images into machine-readable text. OCR (Optical Character Recognition) handles scanned filings, while table extractors preserve the structure of financial statements. Document classification identifies file types, which is essential during due diligence. An agent must distinguish contracts from correspondence to organize a data room effectively. Entity extraction pulls structured data from unstructured text. Extracting a borrower's name, facility amount, and maturity date from a credit agreement enables analysis that otherwise requires manual review.

Computation tools handle tasks requiring calculation rather than lookup. Deadline calculators determine dates based on rules. For example, Federal Rules of Civil Procedure require answers within 21 days of service (Legal Information Institute 2024). Calculating the specific date requires accounting for weekends, holidays, and local rules—a computational task. Citation formatters convert case information into standard styles, such as *The Bluebook*. Financial tools normalize identifiers, mapping between Tickers, CUSIPs (US securities IDs), and ISINs (international IDs). Risk metrics calculate quantitative measures like Value at Risk (VaR) or duration from position data. These computations generate new information for the agent without modifying external portfolios.

4.2 Model Context Protocol (MCP)

The Model Context Protocol (MCP) standardizes how agents access tools (Anthropic 2024). Historically, every database required bespoke integration: Westlaw used one format, Lexis another, and Bloomberg a third. MCP creates a universal interface: learn the protocol once, access any compatible tool. This standardization simplifies governance by providing a single point of audit and control.

The architecture defines three roles. The **MCP Host** manages the agent and controls access, functioning like a firm's IT department determining database subscriptions. The **MCP Client** is the agent-side component that discovers and uses tools. The **MCP Server** exposes capabilities through a standardized interface. A document management system, internal knowledge base, or custom legal research tool can each run as an MCP server.

Communication follows a consistent pattern. Servers publish manifests declaring capabilities. Clients connect through hosts. Clients then send structured requests, and servers return structured results.

For perception specifically, MCP defines **Resources** as read-only data access endpoints. Resources allow agents to:

- Query case law databases and receive structured results;
- Retrieve documents from management systems;
- Access market data feeds;
- Search internal knowledge bases; and
- Fetch regulatory filings.

Resources are explicitly read-only. They implement perception without enabling action. This separation enables fine-grained access control. An agent might have resource access (read documents) without tool access (file documents).

MCP Eliminates the Integration Bottleneck

Without MCP, connecting 10 agents to 10 tools requires 100 custom integrations. With MCP, the same setup requires only 20 implementations, as each agent and tool learns the protocol once.

In legal practice, a single agent queries document management systems, internal knowledge bases, and custom research tools through one protocol. In financial services, one agent accesses portfolio systems, risk engines, and compliance databases through the same standard.

By mid-2025, research benchmarks note that the ecosystem already included *over ten thousand* MCP servers, illustrating the scale and volatility of the integration surface area (Wang et al. 2025).

4.3 Memory as Perception into Institutional Knowledge

Memory systems (Section 6) serve as perception tools for institutional knowledge. When an agent queries a precedent database, it perceives accumulated expertise through specific mechanisms.

Retrieval-Augmented Generation (RAG) enables semantic search over document archives. The agent locates conceptually similar content rather than just keyword matches. A search for "breach of fiduciary duty" retrieves documents about "violation of trust obligations" even if the exact words differ. Vector stores power this search by encoding documents as high-dimensional embeddings (mathematical representations of meaning). This technology enables perception into knowledge bases too large to fit in an agent's active context.

Institutional memory provides access to prior work product. When drafting a new registration statement, an agent can perceive prior S-1 filings (IPO registrations), SEC comment histories, and

successful disclosure language. This access allows current work to build on verified precedents.

Memory-as-perception distinguishes experienced agents from novices. While a junior associate might reason from first principles, a senior associate draws on pattern recognition from hundreds of matters. Memory provides agents with this accumulated experience.

4.4 Domain-Specific Perception Requirements

Perception for regulated professional services requires specialized enhancements. General-purpose search is insufficient; professional agents require authority tracking, jurisdictional awareness, and confidentiality boundaries.

Authority and Verification. Information varies in authority. Perception systems must track provenance to ensure reliability. Authority weighting ranks primary sources (statutes, binding precedent) above secondary sources (law reviews, news). When searching for "insider trading liability," a Supreme Court opinion outranks a commentary article. Source verification confirms that retrieved information originates from the claimed source. Perception tools must return verifiable citations, not just text. Currency validation ensures the authority remains valid. Integrated citators (like Shepard's or KeyCite) verify that retrieved cases have not been overruled.

Jurisdiction and Temporal Scope. Legal and regulatory information is bounded by jurisdiction. California precedent does not bind Texas courts; SEC rules differ from CFTC rules. Perception tools must filter results by relevant jurisdiction. Temporal validity is equally critical. Laws change, and financial data expires. Perception systems must track effective dates. In finance, validity varies by context: milliseconds for trading prices, quarters for compliance reporting. Identifier resolution manages the proliferation of formats. "123 F.3d 456" and "123 F3d 456" refer to the same case. Financial identifiers include tickers, CUSIPs, and LEIs (Legal Entity Identifiers). Perception must normalize these to ensure consistent retrieval.

Matter and Client Isolation. Critically, perception must respect confidentiality boundaries. An agent working on Matter A cannot perceive documents from adverse Matter B. This enforcement of **ethical walls** must occur at the perception layer. In financial contexts, an agent advising Client X cannot perceive material non-public information (MNPI) from Client Y's engagement. Every perception event must be logged, capturing the agent, the query, and the matter context. This audit trail enables compliance review and breach detection. See Section 6 for detailed treatment of isolation requirements.

4.5 Tool Design Principles

Robust perception tools follow design principles that enable reliable operation in professional environments.

Single Responsibility. Each tool should perform one function well. Poorly designed tools bundle multiple functions—searching, formatting, and validation—into opaque interfaces. Untyped return values obscure what callers can expect.

Poor Design: Bundled Functions, Untyped Returns

```
def legal_research(query: str, format: bool,
                   validate: bool, extract: bool) -> dict:
    """Returns... something. Good luck."""
    ...
```

When such a tool fails, diagnosing the error is difficult. A better approach separates tools by function with typed returns. This allows the agent to compose them and isolates failures.

Better Design: Single Responsibility, Typed Returns

```
def search_cases(query: str, jurisdiction: str) -> list[Citation]:
    """Returns matching citations from case law database."""

def retrieve_case(citation: Citation) -> CaseText:
    """Fetches full text for a specific citation."""

def shepardize(citation: Citation) -> CitatorResult:
    """Checks validity: good law, distinguished, overruled."""

def format_citation(case: CaseText, style: str) -> str:
    """Converts to Bluebook, ALWD, or other format."""
```

Graceful Failure. Production systems inevitably fail. Tools should return informative errors rather than generic exceptions. A poor approach raises exceptions that provide no context.

Poor: Opaque Exception

```
def retrieve_case(citation: str) -> dict:
    result = db.query(citation)
    return result["text"]  # raises KeyError if not found
```

A better approach uses typed result objects that make success and failure explicit.

Better: Typed Result with Structured Errors

```
class CaseNotFoundError(BaseModel):
    citation: str
    reason: str
    suggestions: list[str]

def retrieve_case(citation: Citation) -> CaseText | CaseNotFoundError:
    """Returns case text or structured error with recovery options."""
    if not (result := db.query(citation)):
        return CaseNotFoundError(
            citation=str(citation),
            reason="Case may not be in database",
            suggestions=["Check citation format", "Try alternative reporter"
    ]
        )
    return CaseText(...)
```

In professional practice, graceful failure prevents malpractice. When an agent cannot find authority, it must report that explicitly rather than proceeding silently.

Least Privilege and Rate Limiting. Perception tools should request minimum necessary permissions. A legal research tool requires read access to case databases, not write access to the document management system. If a compromised agent gains perception credentials, damage is limited to information disclosure rather than destruction. Rate limiting addresses a common failure mode: infinite search loops. Tools should track invocation frequency and refuse requests beyond reasonable thresholds. If an agent searches five times without results, the tool should force a stop and escalation.

4.6 Evaluating Perception Capabilities

When evaluating agentic systems, you should assess perception against criteria that matter for professional practice.

Coverage determines which sources the agent can access. A litigation agent that queries Westlaw but not state-specific databases has incomplete coverage. You must map available perception tools against information needs to identify gaps.

Retrieval quality measures whether the agent finds relevant information. Test with known-good queries where the correct result is established. Measure both precision (relevance of results) and recall (completeness of relevant results).

Verification confirms that the system distinguishes authoritative from secondary sources. You must ensure that retrieved information is traceable to its source and that citations are independently

verifiable.

Access controls ensure that permissions are appropriate. The agent must access only what it should, and confidentiality boundaries must hold across matter and client lines.

Failure handling reveals system behavior when perception fails. Does it retry, try alternatives, or escalate? It must not crash or proceed with incomplete information.

Audit capability confirms that every perception event is logged. You must be able to reconstruct what information the agent accessed during a task for compliance review.

4.7 From Perception to Action

Perception enables agents to gather information, but professional value requires effecting change: filing documents, sending communications, or executing trades. The distinction between perception and action is fundamental.

Perception tools are read-only. They observe without changing the world. If a perception tool fails or returns incorrect results, no external state has changed; you can retry or attempt alternatives. Action tools, in contrast, change state. They file documents, send emails, and execute trades. Once executed, many actions cannot be undone. The risks differ, and therefore the governance must differ. Section 5 examines action capabilities in detail.

5 How Does an Agent Make Things Happen?

A junior associate's role extends beyond research to producing work product. They draft memos, send emails, file documents, and schedule meetings. A trader's role extends beyond analysis to execution. They enter orders, route trades, and confirm allocations. Value derives from action, not just observation.

Agentic systems face the same imperative. An agent that only reads—searching databases, retrieving documents, analyzing information—produces no deliverable. To complete tasks, agents must *act*: generate documents, send communications, update systems, or execute transactions (Schick et al. 2023). Action implements the "A" in the GPA+IAT framework.

Action Tools

Action tools enable agents to change the state of external systems. Unlike perception tools (read-only), action tools *write*: they file documents, send messages, execute trades, and update databases. Once executed, some actions cannot be undone.

The distinction between perception and action is fundamental to governance. Perception risks involve internal errors (accessing wrong information). Action risks involve external

consequences (harming clients, violating regulations, creating liability).

5.1 Action Tool Categories

Action tools vary in consequence. The critical dimension is **reversibility**: the cost and feasibility of undoing an action. Research on rollback-augmented systems demonstrates that selective state rollback reduces catastrophic failures in safety-sensitive environments (Learning to Undo: Rollback-Augmented Reinforcement Learning with Reversibility Signals 2024). We categorize action tools along a spectrum from easily undone to permanent.

Communication tools send information to others. Internal communications (emails to colleagues, Slack messages) are *partially reversible*. You can follow up with corrections, though you cannot unsend. External communications (emails to clients, letters to counsel) carry higher stakes because recipients are outside your control. Retractions are possible but damage professional reputation. Automated alerts (compliance notifications, reminders) are generally low-risk if templated. Governance typically relies on post-hoc review for internal actions and pre-approval for external ones.

Document management tools create and organize work product. These are *largely reversible*. Drafting memos, generating reports, and filing documents in internal systems occur within the firm's control. Revisions are possible until distribution. Template application (populating standard forms) is low-risk if the templates are pre-validated. The primary control is review before external release.

Filing and submission tools send documents to external authorities. These are *largely irreversible*. Court filings via CM/ECF (Case Management/Electronic Case Files) become public record upon submission. Amendments are possible but the original remains visible. Regulatory submissions via EDGAR (Electronic Data Gathering, Analysis, and Retrieval) or FINRA systems trigger legal obligations. Errors may compel public corrections or enforcement actions. Contract execution creates binding legal obligations that are difficult to unwind. These actions require mandatory pre-approval.

Transaction execution tools transfer value or change ownership. These are *effectively irreversible* or costly to reverse. Trade execution involves entering orders and confirming allocations. Reversal requires offsetting trades at market prices, realizing any loss. Payment processing (wire transfers) moves funds immediately; recovery relies on recipient cooperation. System updates (modifying production databases) can disrupt operations. These actions demand the strictest controls: multi-factor approval, segregation of duties, and real-time monitoring.

5.2 The Reversibility Framework

Reversibility dictates oversight structure. Consider how you delegate to a junior associate.

For **fully reversible** actions (research, drafting), the associate works independently. You review the output before it matters. Errors are internal and cost-free to fix.

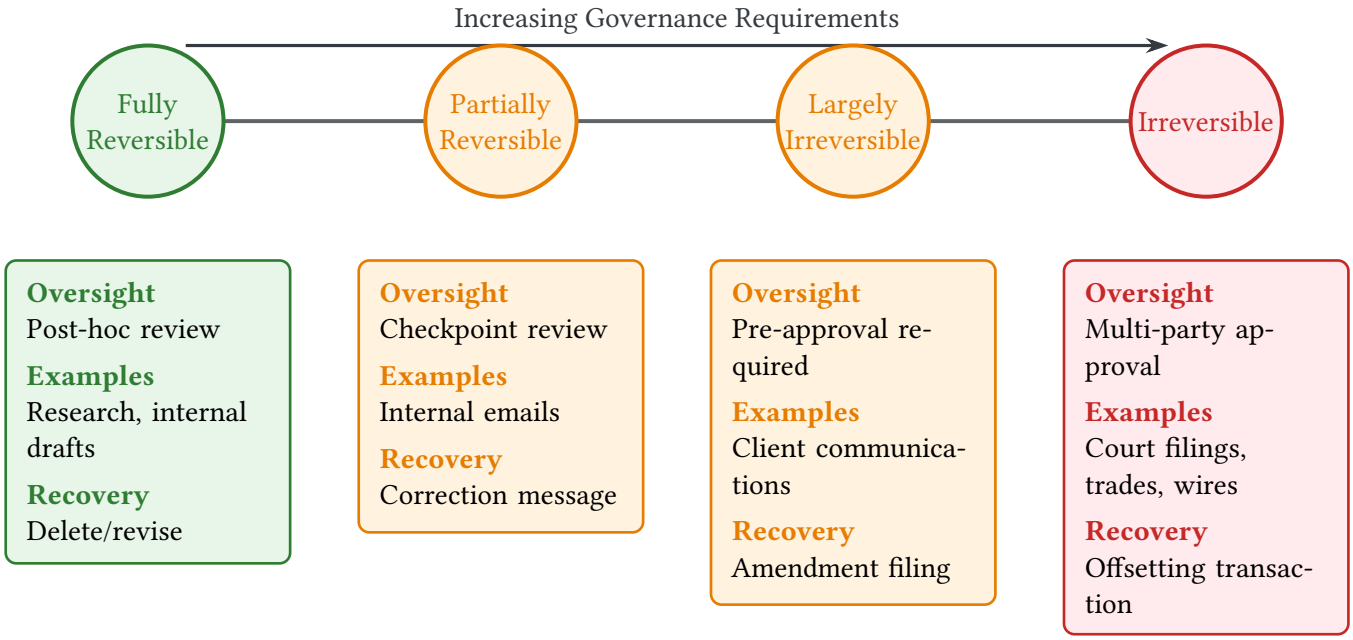

Figure 3: Action reversibility spectrum and corresponding governance requirements. As actions become less reversible, oversight shifts from post-hoc review to pre-approval and multi-party authorization. Recovery mechanisms range from simple deletion for fully reversible actions to complex offsetting transactions for irreversible ones.

For **partially reversible** actions (internal emails, file organization), you employ checkpoint review. The associate completes the work, and you review it periodically or before a major milestone.

For **largely irreversible** actions (client emails, filings), you require pre-approval. The associate prepares the draft, but you must approve it before execution.

For **irreversible** actions (trades, wires), you implement multi-party approval. No single individual—human or machine—can execute alone.

Agent governance must track this framework. The architecture should enforce controls corresponding to the action's reversibility classification.

5.3 MCP Tools and Prompts for Action

The Model Context Protocol (MCP) defines two capability types relevant to action governance.

MCP Tools are executable functions that change state. Unlike read-only Resources, Tools create documents, send communications, submit filings, and execute transactions. Tool manifests should include risk metadata: reversibility classification, approval requirements, and audit logging needs. This allows the MCP Host to enforce controls automatically.

MCP Prompts are reusable templates for common tasks. For action workflows, prompts encode Standard Operating Procedures (SOPs).

Table 2: Action reversibility and required oversight

Reversibility	Examples	Oversight	Recovery
Fully reversible	Research, internal drafts, calculations	Post-hoc review	Delete/revise
Partially reversible	Internal emails, document filing, alerts	Checkpoint review	Correction/follow-up
Largely irreversible	Client communications, court filings, regulatory submissions	Pre-approval required	Amendment/retraction (visible)
Irreversible	Trade execution, wire transfers, contract execution	Multi-party approval	Offsetting transaction (costly)

- **Legal:** Contract review checklists, filing preparation workflows.
- **Finance:** Trade compliance checks, client onboarding sequences.

Prompts standardize action sequences, reducing variation and error. They act as "guardrails" by ensuring the agent follows the approved procedure for high-stakes actions.

5.4 Action Security

Every action interface is a security boundary. Actions access external systems and create real-world consequences.

All action tools must implement core security controls:

- **Authentication:** Verify the agent's identity via service accounts with strong credentials.
- **Authorization:** Enforce role-based access control (RBAC) and least privilege.
- **Input Validation:** Reject malformed requests by validating all parameters against strict schemas.
- **Output Confirmation:** Require human approval before executing high-stakes actions.
- **Rate Limiting:** Cap action frequency to prevent runaway execution.
- **Audit Logging:** Record every action with context (agent, timestamp, parameters, matter/client).

Beyond core controls, specific threats require targeted mitigations (OWASP Foundation 2025; Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents 2024):

Prompt Injection via Action Parameters. Adversaries may embed malicious instructions in data that the agent processes and passes to tools. Mitigation requires sanitizing all parameters and never passing raw user input directly to sensitive action interfaces.

Privilege Escalation via Tool Chaining. An agent might combine multiple low-privilege tools to achieve a high-privilege outcome. Mitigation involves analyzing tool combinations and requiring

approval for sequences that cross security boundaries.

Action Replay. An attacker might capture a valid action request (e.g., "Pay $100") and replay it multiple times. Mitigation requires **nonces** (unique, one-time numbers) or timestamps to ensure each request is processed only once.

5.5 Approval Workflows

For non-reversible actions, the agent prepares and the human approves (Evaluating Human-AI Collaboration: A Review and Methodological Framework 2024). We define three patterns for this division of responsibility.

The **Single Approver** pattern suits routine actions with clear authority. The agent completes preparation and presents it to one designated human. *Example:* The agent prepares a draft court filing; the supervising attorney reviews and approves; the agent submits.

The **Multi-Party Approval** pattern applies to high-stakes actions with significant exposure. Multiple independent humans must sign off. *Example:* The agent prepares a wire transfer. Operations reviews the amount. Compliance reviews the purpose. A manager provides final approval. Only then does the agent execute.

The **Escalating Approval** pattern adjusts authority based on risk tiers. *Example:* Trades under $100k require desk manager approval. Trades between $100k and $1M require senior trader approval. Trades over $1M require CIO approval.

Effective approval requests must enable informed decision-making. The agent should present:

- **Action:** Clear description of what will happen.
- **Context:** Why is this needed?
- **Risk:** What are the potential negative outcomes?
- **Reversibility:** Can this be undone?
- **Evidence:** What data supports this decision?

The approver should be able to decide based on the request alone, without needing to investigate the raw data.

5.6 Rate Limiting and Circuit Breakers

Agents can fail in loops: repeatedly submitting the same request, sending duplicate messages, or retrying failed transactions.

Rate Limiting caps action frequency. The thresholds below are *illustrative*; calibrate them to your workflow, risk tolerance, and regulatory obligations.

- *Per-Action:* e.g., max 5 emails per minute.
- *Per-Matter:* e.g., max 20 actions per day without review.

- *Cost:* e.g., max $1,000 in transaction fees per session.

When limits are reached, the agent must pause and escalate.

Circuit Breakers automatically stop execution upon anomaly detection. These examples are also *illustrative* and should be tuned to baseline behavior and threat models.

- *Failure Count:* e.g., if an action fails three times, stop.
- *Spike Detection:* e.g., if action rate spikes 5× above baseline, pause (potential compromise).
- *Cumulative Limit:* If daily total exceeds safety thresholds, lock the system.

Circuit breakers transform runaway failures into controlled pauses, buying time for human intervention.

5.7 Evaluating Action Capabilities

When evaluating agentic systems, assess action capabilities against professional standards.

Inventory: Map all available action tools against workflow requirements. Verify that no "unnecessary" tools are enabled (Least Privilege).

Classification: Verify that each tool is correctly classified by reversibility. Ensure that "delete database" is not classified as "fully reversible."

Controls: Confirm that approval gates exist for all irreversible actions. Verify that approvers receive sufficient context. Check authentication and audit logging.

Safety: Test rate limits and circuit breakers. Simulate a "runaway agent" scenario to ensure the system locks down. Verify rollback procedures: if an action fails, can you recover?

5.8 From Action to Governance

Action tools are where agentic systems create real-world consequences. Governance here differs in kind from perception governance.

Perception risks (accessing wrong data) are internal. Action risks (sending wrong emails, executing wrong trades) are external and potentially irreversible. This section established the architectural controls: the Reversibility Framework, approval workflows, and circuit breakers.

Section 9 examines when agents should *not* act—recognizing situations that require human decision-making. The interplay between action capability and escalation judgment is central to safety. Section 6 addresses memory: how agents maintain context across sessions and learn from experience.

6 How Does an Agent Remember Things?

The previous two sections examined how agents gather information (perception) and effect change (action). However, these capabilities operate in the moment: the agent perceives the current state, reasons about it, and acts. Without memory, every interaction resets. The agent cannot recall prior research, successful strategies, or case history. Memory transforms an agent from a stateless tool into a system that learns and adapts.

Every experienced professional knows that institutional memory distinguishes efficient work from reinventing the wheel. When you start a new securities registration, you do not begin from scratch. You pull prior S-1 filings (IPO registrations), review SEC comment history, and check the precedent database for disclosure language. You do not re-research basic questions. The firm maintains templates that incorporate years of accumulated knowledge.

The same principle applies to portfolio management. When you revisit an equity position, you do not rebuild the investment thesis. You retrieve the research file, review prior models, and update assumptions with new data. Accumulated research enables incremental refinement rather than duplicative work.

Memory in agentic systems serves a parallel function: context retention across sessions and learning from experience. With memory, the agent maintains continuity—building on prior work rather than starting over.

Agent Memory

Agent memory stores and retrieves information across timescales (Park et al. 2023; Wang et al. 2023).

- **Working Memory:** The documents actively loaded in the agent's context (like papers on a desk).
- **Episodic Memory:** The history of actions and outcomes for a specific matter (like a case file).
- **Semantic Memory:** The general principles and institutional knowledge available for retrieval (like the firm's precedent archive).

6.1 Memory Types: From Desk to Archive

Law firms use layered filing systems, each suited to different timescales. The associate's desk holds active work; the matter file contains engagement history; the precedent database archives institutional knowledge. Each layer trades immediacy for capacity. Agent memory systems follow the same pattern.

Working memory utilizes the **context window**—the text currently loaded in the LLM's active attention. Just like desk space, context windows have strict limits. An associate can only have so many documents open at once; an agent can only hold so many **tokens** (units of text, roughly 0.75

words) in active context. As of late 2025, leading models handle roughly 200,000 tokens. When the task exceeds this limit, you need other storage systems. In finance, this parallels the active trading screen: live prices and positions are immediate but transient.

Episodic memory corresponds to the matter file. Every memo, correspondence, and research result related to an engagement goes here. The associate does not re-research answered questions; the file provides the history. In agentic systems, episodic memory captures the log of actions and outcomes (Park et al. 2023). The agent records: "I searched for Ninth Circuit venue cases, found three opinions, and drafted the analysis." When asked a follow-up, the agent retrieves that prior state. This mirrors the financial research file: you pull prior analysis and update it, rather than starting fresh.

Semantic memory is the firm's precedent database. Institutional knowledge accumulates over decades. When you need a force majeure clause, the database offers fifty examples. Agentic systems implement this through **Retrieval-Augmented Generation (RAG)**: dynamically fetching relevant information from a large corpus to augment reasoning (Lewis et al. 2020). RAG enables agents to access knowledge beyond the context window.

The technology powering RAG is the **vector store**, which makes databases searchable by meaning rather than just keyword. Vector stores encode documents as **embeddings**—high-dimensional numerical representations of semantic meaning (Reimers and Gurevych 2019). A search for "breach of fiduciary duty" finds documents about "violation of trust obligations" because their numerical representations are close in vector space.

6.2 Retrieval-Augmented Generation (RAG)

RAG enables agents to access institutional knowledge, equivalent to asking the firm librarian for "our best research on this issue." While traditional keyword search misses concepts phrased differently, semantic search finds conceptually similar content.

The RAG pipeline has four steps:

1. **Chunking:** Breaks documents into semantic units (e.g., paragraphs or sections) while preserving metadata (source, date).
2. **Embedding:** Converts each chunk into a vector that encodes its meaning.
3. **Retrieval:** Finds chunks similar to the user's query by comparing vectors.
4. **Generation:** Injects the retrieved content into the agent's prompt, allowing the LLM to answer using the specific knowledge.

Robust implementations enhance basic RAG with advanced patterns. **Hybrid retrieval** combines semantic search (vectors) with keyword search (BM25), catching both conceptual matches and exact terms (like specific case citations). **Query rewriting** transforms vague user questions ("What's the

rule?") into specific search queries based on conversation history. **Reranking** scores initial results by authority or relevance, ensuring binding precedent ranks above secondary sources. **Filtered retrieval** constrains results by metadata, such as jurisdiction or date, preventing New York agents from citing California law.

Warning: The Hallucination Risk

Fabricated citations are a critical failure mode. Studies show that even RAG-enabled tools can hallucinate 17–33% of the time (Dahl et al. 2024). You must implement a strict verification step: before any citation reaches the user, the system must verify that the source exists in the retrieved context.

6.3 Domain-Specific Memory Considerations

Memory for regulated services requires enhancements beyond generic implementations. You must account for authority, jurisdiction, and time.

Authority Weighting. Not all information is equal. Perception systems must rank primary authority (statutes, binding precedent) above secondary sources. When searching "insider trading liability," a Supreme Court opinion outranks a law review article. Financial systems effectively rank SEC no-action letters above client alerts.

Jurisdiction Awareness. Legal information is bounded by jurisdiction. California precedent does not bind Texas courts. Metadata tagging must enable strict filtering. An agent researching Delaware corporate law must not surface New York cases as controlling authority.

Temporal Validity. Law and markets change. Citator integration (like Shepard's) validates that cases remain good law. A 1985 case may be overruled; RAG must surface the current state. In finance, validity varies by context: trading data expires in milliseconds, while industry analysis lasts quarters. Memory systems must tag data with effective dates and trigger refreshes when content becomes stale.

Identifier Resolution. Citations and tickers vary. "123 F.3d 456" and "123 F3d 456" are the same case. Companies have multiple identifiers: Ticker, CUSIP, ISIN, LEI. Without normalization, retrieval fails to connect related records.

6.4 Matter and Client Isolation

Critically, memory systems must enforce **ethical walls**. Information from Matter A cannot leak into Matter B. If an agent uses Matter A's privileged info on adverse Matter B, it risks privilege waiver and malpractice. In finance, this prevents Material Non-Public Information (MNPI) leakage.

Implementing separation requires four controls:

- **Namespaces:** Give each matter an isolated memory partition.
- **Scoped Retrieval:** Queries strictly access only the current matter's namespace.
- **Access Control:** Role-based permissions determine which agents (and humans) can access a namespace.
- **Audit Trails:** Log every read/write with timestamp, identity, and matter ID for compliance review.

Secure deletion and retention controls are often required by organizational policy and may be legally required depending on the engagement, jurisdiction, and applicable retention rules. When a matter closes or a client relationship ends, memory should be deleted or archived according to a documented retention schedule, with deletion (when required) implemented in a verifiable way.

6.5 Evaluating Memory Systems

Testing memory requires specific metrics for professional practice.

Retrieval Quality. Measure *precision* (are retrieved documents relevant?) and *recall* (did we find all relevant documents?). Use "gold standard" research memos as test sets: does the agent find the same authorities the human expert cited?

Isolation Integrity. Verify that cross-matter queries return zero results. Matter A's agent should never retrieve Matter B's documents. Red-team testing should deliberately attempt to breach these walls.

Temporal Validity. Measure freshness. How often does the system surface overruled precedent or stale prices? Track the lag between a regulatory change and its availability in the knowledge base.

Scale Performance. Episodic memory grows rapidly. Ensure retrieval latency remains low even as a litigation matter generates thousands of documents.

6.6 From Memory to Planning

Memory provides the context agents need to plan. Without memory, agents repeat failed strategies. Research starts from scratch. Preferences are forgotten.

With memory, agents learn. The agent recalls: "Last time, broad search terms yielded too much noise. This time, I will use narrow queries." Prior work is built upon. User preferences persist.

Memory supports **Adaptation**—the "A" in the GPA+IAT framework—because behavior improves as the system learns from experience.

Section 7 examines the next question: how does an agent break a big job into steps? Just as the case file enables strategic litigation planning, agent memory enables systematic task decomposition.

7 How Does an Agent Break a Big Job into Steps?

A litigation partner approaching a new matter does not start by drafting motions. The partner develops a strategy: discovery first (identifying needed facts), then dispositive motions if the law favors the client, followed by settlement discussions or trial preparation. Discovery breaks into phases: initial disclosures, document requests, interrogatories, and depositions. Tasks distribute across the team: a senior associate handles briefing, a junior associate reviews documents, and a paralegal manages scheduling. Throughout, the partner monitors progress against deadlines and adjusts strategy based on new facts.

This is **planning**: decomposing complex goals into action sequences. It mirrors the litigation roadmap or deal timeline that guides execution. Without planning, agents react to immediate observations without strategy. With planning, they work systematically toward objectives, adapt when circumstances change, and recognize completion.

Planning

Planning decomposes complex goals into sequences of actions. It encompasses:

- **Decomposition**: Breaking large tasks into manageable steps.
- **Sequencing**: Ordering steps logically based on dependencies.
- **Allocation**: Assigning steps to specific tools or sub-agents.
- **Monitoring**: Tracking progress toward the goal.
- **Adaptation**: Adjusting the plan when circumstances change.

Without planning, an agent resembles an associate running searches without a strategy—busy but not progressing toward a deliverable.

7.1 Planning Patterns

Three patterns dominate agent planning, each suited to different task types.

ReAct (Reasoning + Acting). The most fundamental pattern interleaves reasoning with action (Yao et al. 2022). Consider a partner asking for authority on an unenforceable forum selection clause. The associate reasons: "Key grounds are unconscionability and public policy. I will start with *Atlantic Marine*." They search, observe results, and reason again: "Unconscionability cases involve consumer contracts, not our commercial context. The public policy line is stronger." They search again, refining based on results.

Each cycle has three components:

- **Thought**: Explicit reasoning about what to do next.
- **Action**: A tool call to gather information or effect change.
- **Observation**: The tool output that informs the next thought.

Reasoning traces make decisions transparent and auditable. ReAct works well for exploratory tasks where you learn as you go—legal research, fact investigation, and market analysis.

Plan-Execute. This pattern separates planning from execution. For a document review task ("Review 50 contracts for choice-of-law provisions"), the associate creates a plan: list the contracts, open each one, extract the provision, and record findings. Then they execute systematically. The plan remains static because the task is well-defined.

Plan-Execute fits established workflows: due diligence checklists, compliance reviews, and document assembly. You create the plan upfront and execute methodically. Research variants like ReWOO (Xu et al. 2023) (separating reasoning from observation) and LLMCompiler (Kim et al. 2024) (optimizing execution graphs) enable parallel tool calling. However, the core pattern remains: plan first, then execute.

Hierarchical Planning. Law firms decompose matters into workstreams delegated through layers. A parent agent receives a high-level goal, breaks it into sub-goals, and delegates to specialists. "Prepare for trial" becomes:

- Finalize witness list (delegated to Agent A).
- Prepare exhibits (delegated to Agent B).
- Draft jury instructions (delegated to Agent C).

Each specialist may decompose further. This enables parallelization and specialization, mirroring how litigation teams work. Section 10 details these coordination patterns.

These patterns represent a shift from traditional workflow automation. Traditional engines used **static orchestration**: predefined graphs specifying exact steps and branches (BPM systems). The workflow was designed at build time; the engine merely executed it.

Static vs. Dynamic Orchestration

Static orchestration executes predefined workflow graphs. The same input always produces the same execution path. It is predictable and auditable but inflexible.

Dynamic orchestration reasons about task decomposition at runtime. The LLM examines the goal, considers available resources, and constructs a plan on the fly. It is adaptive but less predictable.

LLM-based orchestration is inherently dynamic. "Prepare for trial" decomposes differently depending on case complexity. The LLM constructs a delegation structure based on the specific context. This adaptability is both the promise and the challenge.

Static workflows handle anticipated scenarios. Dynamic orchestration handles novel situations. Maintenance costs also differ: static workflows require explicit updates, while dynamic orchestration absorbs changes through prompt updates.

Auditability Challenge

Static workflows produce predictable execution traces. Dynamic orchestration may produce different decompositions for similar inputs, complicating audit. For regulated applications, you must log the *reasoning* behind orchestration decisions, not just the decisions themselves.

Production systems often combine both. High-volume, well-understood processes use static workflows. Complex, novel tasks use dynamic orchestration. The planning patterns described above—ReAct, Plan-Execute, Hierarchical—are forms of dynamic orchestration.

7.2 Choosing the Right Planning Pattern

Selecting the right pattern depends on task structure and required autonomy.

Table 3: Planning pattern selection guide

Task Type	Pattern	Autonomy	Example
Well-defined steps, known scope	Plan-Execute	Moderate	Credit review, compliance audit, due diligence checklist
Exploratory, learns as it goes	ReAct	Higher	Legal research, fact investigation, market analysis
Complex, parallel workstreams	Hierarchical	Distributed	M&A transaction, portfolio construction, multi-jurisdiction filing

Higher autonomy requires more sophisticated oversight.

Plan-Execute operates with moderate autonomy. The agent works within bounds defined by the plan. Oversight focuses on plan validation. ReAct involves higher autonomy because the agent decides what to search and when to stop. Oversight requires explicit termination mechanisms and confidence thresholds. Hierarchical patterns distribute autonomy. Oversight requires clear delegation contracts and escalation paths. You must match oversight rigor to autonomy level.

7.3 Understanding the Task Before Planning

Before planning, agents must understand the request. Section 3 covers intent extraction. For planning, the key outputs are task classification, constraints, and success criteria.

Task classification determines the planning pattern: exploratory (ReAct), structured (Plan-Execute), or complex (Hierarchical). Constraints define the bounds: deadlines, budgets, and scope. Success criteria define how the agent recognizes completion.

Effective planning requires clear inputs. Ambiguous goals produce unfocused plans. Unclear success criteria make termination difficult.

7.4 Budget Architecture

Without resource budgets, agents can run indefinitely. This is the "runaway associate" problem: asking for two cases and receiving fifty. We allocate budgets as a planning mechanism to control execution.

Four budget types control consumption:

- **Token Budgets:** Limit LLM API consumption to prevent expensive reasoning loops.
- **Time Budgets:** Enforce deadlines by stopping execution after a fixed duration (e.g., 10 minutes).
- **Tool Call Budgets:** Cap external interactions (e.g., max 20 searches).
- **Cost Budgets:** Cap total spending in dollars.

Budgets cascade. A session budget constrains the engagement; a task budget allocates resources to items; subtask budgets subdivide further. A research task might receive 30 minutes and 50,000 tokens. Subtasks share this pool rather than receiving unlimited resources.

Token costs compound. A 200-page credit facility requires roughly 80,000 tokens to ingest. Analysis consumes more. A comprehensive review might reach 1,000,000 tokens. You must monitor aggregate costs, not just per-task.

The economics depend on the task. Retrieval-heavy tasks (document review) show clear ROI. Judgment-intensive tasks require extensive human review, potentially reducing ROI. Transparency about AI assistance enables clients to evaluate value (American Bar Association Standing Committee on Ethics and Professional Responsibility 2024).

Agents should degrade gracefully when budgets tighten. Tiered outputs provide value at every level. A minimal budget returns the statute. A moderate budget adds holdings. A full budget delivers analysis. Soft limits (80% of budget) warn the agent to prioritize completion. Hard limits (100%) terminate execution. A budget-aware agent that delivers partial results is superior to one that fails completely.

7.5 Knowing When to Stop

Knowing when to stop is a critical planning capability. We define four stopping conditions:

1. **Success:** The goal is achieved (question answered, document reviewed).
2. **Resource Exhaustion:** Budgets are hit. The agent returns partial results or escalates.
3. **Confidence Thresholds:** Uncertainty is too high. The task routes to human review.
4. **Error Conditions:** Repeated failures indicate a problem retrying won't solve.

You must define explicit stopping rules. Instruct the agent as you would an associate: "If you find three consistent opinions, stop. If you find nothing after two hours, escalate."

7.6 Guardrails and Loop Detection

Agents can get stuck in loops despite budgets. Mechanisms like step limits, reflection checkpoints, and external watchdogs prevent this. Section 8.4 examines these in detail.

7.7 From Planning to Termination

Planning decomposes work, but every plan must end. The next two questions address the boundaries of autonomous execution.

Section 8 (Termination) addresses how an agent knows it is done. This requires defining success criteria and budget limits. Section 9 (Escalation) addresses how an agent knows when to ask for help. This requires confidence thresholds and authority boundaries.

Without clear termination, agents run forever. Without escalation, they exceed authority. These boundaries define the safe operating envelope for autonomous systems.

8 How Does an Agent Know When It's Done?

Every professional learns to recognize completion. The research memo is done when you have found sufficient authority and synthesized it. The due diligence is done when you have reviewed all material documents. The trade is done when the order executes and settles. Knowing when work is complete distinguishes effective professionals from those who over-research or under-deliver.

Agents face the same challenge. Without explicit termination conditions, agents can run indefinitely: searching one more database, trying one more approach, refining one more time. We call this the "runaway associate" problem: you ask for two relevant cases, and the associate gives you fifty because they did not know when to stop.

Termination

Termination conditions define when an agent should stop executing. Three outcomes are possible:

- **Success:** The goal is achieved, and the agent delivers the result.
- **Failure:** The goal cannot be achieved; the agent reports why and stops.
- **Escalation:** The agent cannot determine success or failure, transferring the decision to human judgment.

Termination implements the "T" in the GPA+IAT framework. Without it, agentic systems lack the property that distinguishes systems from runaway processes.

8.1 Termination Condition Categories

Five categories of termination conditions bound agent execution: success conditions, resource budgets, confidence thresholds, error conditions, and escalation triggers.

Success conditions trigger when the goal is achieved.

- **Completeness:** Have all checklist items been addressed? (e.g., all 50 contracts reviewed).
- **Quality:** Is the output sufficient? (e.g., conclusions supported by binding authority). This often requires human validation.
- **Convergence:** Has the agent stopped learning? If three consecutive searches yield no new authority, the research is likely saturated.

Resource budgets provide hard limits to prevent runaway execution. Section 7.4 details budget architecture; here we focus on termination.

- **Token budgets:** Stop after 50,000 tokens to prevent expensive reasoning loops.
- **Time budgets:** Enforce deadlines (e.g., stop after 10 minutes).
- **Iteration budgets:** Cap tool calls (e.g., max 20 searches).
- **Cost budgets:** Halt after spending a fixed dollar amount.

Budgets cascade. A task might hit a time limit before exhausting tokens. Budget exhaustion is not necessarily failure; partial results are often valuable.

Confidence thresholds gate actions on certainty. When confidence is high, the agent delivers the answer. When confidence drops below a threshold (e.g., 80%), the agent stops and escalates. This mirrors associate behavior: "I'm not confident. Let me ask the partner." Calibrating these thresholds is challenging (Kadavath et al. 2022). Agents can be overconfident. Effective calibration requires testing against known outcomes.

Error conditions require agents to recognize failure modes.

- **Repeated failures:** If Westlaw times out three times, stop rather than retrying indefinitely.
- **Inconsistent data:** If the 10-K revenue differs from the earnings release, stop and flag for human review.
- **Constraint violations:** If a planned action exceeds position limits, stop immediately.
- **Impossibility:** If requirements conflict, report the impossibility rather than compromising.

Escalation triggers require human judgment regardless of success or failure. Novel situations, high-stakes decisions, and actions exceeding authority boundaries must trigger termination and handoff. Section 9 examines these patterns.

8.2 Defining Success Criteria

Vague goals produce unclear termination. "Research the statute of limitations" is ambiguous. Effective success criteria provide clear signals.

Completeness checklists enumerate deliverables. For a credit agreement review, the checklist requires identifying financial covenants, comparing them to market terms, and summarizing risks. The agent terminates only when all items are complete.

Sufficiency thresholds define "enough." For case research, sufficiency might mean finding three on-point circuit opinions. The agent stops upon reaching this count, without searching every database.

Convergence criteria recognize diminishing returns. If consecutive searches yield no new results, the task is likely done.

Deliverable specifications define output format. "A two-page memo with executive summary" tells the agent exactly what success looks like.

Instruct agents as you would an associate: "If you find clear Ninth Circuit authority, you are done. If circuits split, map the split. If you find nothing after two hours, stop."

8.3 Recognizing Failure

Agents must recognize and report failure honestly. Negative results are valuable information. "I searched all databases and found no authority" is a valid finding.

Diagnostic reporting explains the failure. Instead of "Task failed," the agent should report: "I searched Westlaw and Lexis using [queries]. Zero results suggest the issue is novel or terms are wrong. Recommend manual review." This is actionable.

Partial completion must be preserved. If an agent analyzed 4 of 10 articles before failure, it should report: "Articles 1-4 analyzed. Articles 5-10 remain." Preserving partial work prevents wasted effort.

Root cause identification aids the human response. Was it a tool failure (transient) or impossible requirements (structural)? The agent's diagnosis informs the next step.

8.4 Guardrails and Loop Detection

Agents can get stuck in unproductive loops despite termination conditions. We use three mechanisms to prevent this.

Step limits serve as a final backstop: after N steps, stop and require approval. This prevents unbounded execution.

Progress detection monitors value. If the last five actions produced no new information, the agent is likely stuck. This should trigger reflection or escalation.

Reflection steps build self-assessment. The agent asks: "Am I making progress? Should I change approach?"

External watchdogs monitor from outside. If the same tool is called repeatedly with identical

parameters, the watchdog intervenes. Without loop detection, agents will eventually get stuck in production.

8.5 The Reliability Cliff

Benchmarking reveals a sharp reliability boundary. METR (Model Evaluation and Threat Research) tested agents across standardized tasks.

The Four-Minute Cliff

METR's 2025 study found that agents achieve **near-perfect success on tasks under 4 minutes**, but **under 10% success on tasks over 4 hours** (METR 2025).

This gap defines the current boundary for reliable deployment. You must decompose tasks aggressively, keep agent tasks short, and insert human checkpoints. Do not expect autonomous completion of multi-hour workflows.

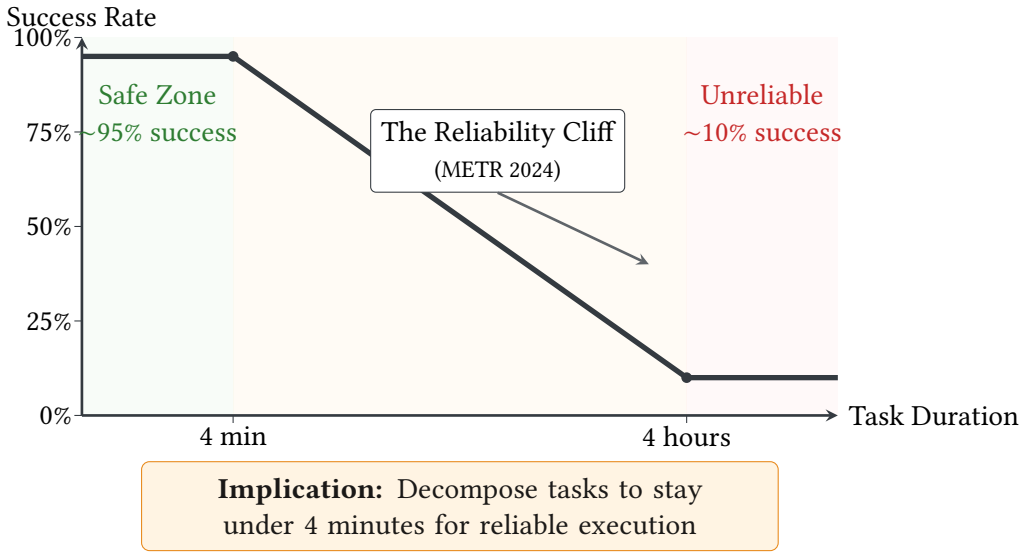

Figure 4: The Reliability Cliff: Agent success rates drop dramatically as task duration increases. METR (2024) found that agents maintain approximately 95% success on tasks under 4 minutes, but success rates fall to roughly 10% for tasks exceeding 4 hours. This sharp transition motivates the decomposition of complex tasks into shorter subtasks and the use of aggressive timeout policies in agent architectures.

This cliff stems from **compounding errors**. A 95% accurate step followed by a 90% accurate step yields 85% reliability. Over dozens of steps, failure becomes a statistical certainty. **Planning fragility** and **integration brittleness** (API failures) further degrade long-running tasks. Design for this reality: decompose, validate, and assume failure.

8.6 Graceful Degradation

When termination occurs early, agents should degrade gracefully.

Tiered outputs provide value at any budget.

- **Low Budget:** Deliver controlling statute and citation.
- **Medium Budget:** Add key holdings.
- **Full Budget:** Deliver comprehensive analysis.

Partial results allow the user to decide if further investment is warranted.

Progress preservation saves state. If an agent stops mid-review, the human should be able to resume without restarting.

Clear status reporting is essential: "Completed 60% of task. Remaining: Articles 5-8. Findings so far: [summary]." The agent should recommend next steps: "Recommend allocating 30 more minutes to complete." This enables informed human decision-making.

8.7 Evaluating Termination Capabilities

Assess termination against six criteria:

- **Success Clarity:** Are termination conditions explicit? Can you predict when it stops?
- **Budget Enforcement:** Does the agent actually stop at the limit?
- **Loop Detection:** Does it detect and break infinite loops?
- **Failure Reporting:** Are error messages actionable?
- **Graceful Degradation:** Does it return useful partial results?
- **Escalation Handoff:** Does the human receive sufficient context to take over?

8.8 From Termination to Escalation

Termination defines when agents stop. Success means the task is done. Failure means it is impossible. Escalation means the agent needs help.

The third category is critical. An agent finding conflicting authority or exceeding authorization must stop and ask for input. This is not failure; it is safety.

Section 9 examines this question: when should an agent stop autonomous operation and ask for help? Termination and escalation together define the boundaries of autonomous execution. Without them, agents run forever or exceed authority.

9 How Does an Agent Know When to Ask for Help?

The best junior associates know when to go to the supervisor. They do not interrupt the partner with every question, but they also do not proceed confidently into territory beyond their expertise.

They recognize authority boundaries: "I can draft this motion, but I need partner review before filing." They recognize competence limits: "I have researched for two hours and cannot find clear authority—I should ask someone with more experience." They recognize high-stakes situations: "The client is asking about strategy, not just research—this needs partner involvement."

Agentic systems require the same judgment. An agent that never escalates will eventually exceed its competence, authority, or the bounds of safe autonomous operation. An agent that escalates everything provides no value: it becomes a complicated way to route work to humans. The challenge is drawing the line.

Escalation

Escalation transfers control from the agent to a human when autonomous execution should stop. Unlike termination, which ends the task (success or failure), escalation pauses the task and requests human input before continuing.
This reflects professionalism, not failure. Recognizing when you need help and asking for it is exactly what we expect from junior professionals. Agents should do the same.

9.1 When to Escalate

Three categories of triggers warrant escalation: mandatory triggers, confidence-based triggers, and error detection (Human-in-the-loop machine learning: a state of the art 2022; Human-AI collaboration is not very collaborative yet: a taxonomy of interaction patterns in AI-assisted decision making from a systematic review 2024).

Mandatory triggers require human involvement regardless of the agent's confidence.

- **Budget Exhaustion:** When resource limits approach, escalate with a progress summary rather than stopping silently.
- **High-Stakes Actions:** Court filings, client communications, and large trades require human approval (approval gates).
- **Authority Boundaries:** Actions exceeding authorized thresholds require human sign-off.
- **Irreversibility:** Actions that cannot be undone (filing, trading) warrant pre-execution review.

Confidence-based triggers occur when uncertainty exceeds acceptable thresholds (Using AI Uncertainty Quantification to Improve Human Decision-Making 2023).

- **Low Output Confidence:** "I found conflicting circuit authority and cannot determine the rule."
- **Conflicting Information:** Data sources disagree (e.g., 10-K vs. earnings release).
- **Novel Situations:** Unprecedented fact patterns or market conditions require human expertise.
- **Ambiguity:** The agent remains uncertain about instructions despite clarification attempts.

Error and anomaly detection triggers escalation when the agent cannot fix the problem.

- **Repeated Failures:** If a tool fails three times, report unavailability rather than retrying.
- **Data Anomalies:** Figures that do not reconcile or suspicious filing dates warrant investigation.
- **Constraint Violations:** If an action would violate policy (e.g., position limits), escalate for confirmation.
- **Impossibility:** Report when the task cannot be completed as specified.

9.2 How to Escalate

Effective escalation provides the human with everything needed to make a decision. A complete handoff includes five parts:

1. **Situation Summary:** Brief context for the human.
2. **Progress to Date:** What has been done and what remains.
3. **Trigger:** Why the agent is escalating now.
4. **Information Gathered:** Relevant findings (even if partial).
5. **Recommendation:** A proposed path forward with reasoning.

Example: Legal Research Escalation.

> **Situation**: Researching statute of limitations for Section 10(b) securities fraud claim.
>
> **Progress**: Searched Westlaw and Lexis. Found clear authority on the 2-year discovery period but conflicting authority on the trigger event.
>
> **Trigger**: Low confidence. The Ninth and Second Circuits apply different tests for inquiry notice. I cannot determine which applies to our facts.
>
> **Findings**: [Summary of key cases with citations]
>
> **Options**: (a) Apply the conservative test; (b) research district court authority; (c) seek partner guidance on the fact pattern.
>
> **Recommendation**: Option (c). The question is fact-intensive, and partner judgment is required.

Example: Financial Escalation.

> **Situation**: Executing rebalancing trades to reduce tech exposure from 35% to 25%.
>
> **Progress**: Generated trade list. Compliance check passed. Ready to execute.
>
> **Trigger**: Trade size ($500K) exceeds single-approver threshold.
>
> **Findings**: Trades realize $45K in short-term gains and $12K in losses. Net tax impact: $8K liability.

Options: (a) Approve full list; (b) prioritize tax-loss positions; (c) execute in tranches.

Recommendation: Option (a) if reducing exposure is urgent; Option (b) if tax optimization is priority.

9.3 Human-in-the-Loop Patterns

We define five patterns for integrating human oversight, suited to different risk profiles.

Approval Gates separate preparation from authorization. The agent drafts; the human approves. This is essential for irreversible actions (filings, trades).

Checkpoint Reviews verify direction at milestones. A research agent presents authorities before drafting, preventing wasted effort.

Confidence-Based Escalation ties autonomy to certainty. High confidence triggers autonomous execution; low confidence triggers escalation.

Human-as-Tool treats human expertise like any other resource. The agent queries an expert when needed, incorporates the response, and proceeds.

Reversibility Classification matches oversight to risk. Fully reversible actions proceed autonomously; irreversible actions require pre-approval.

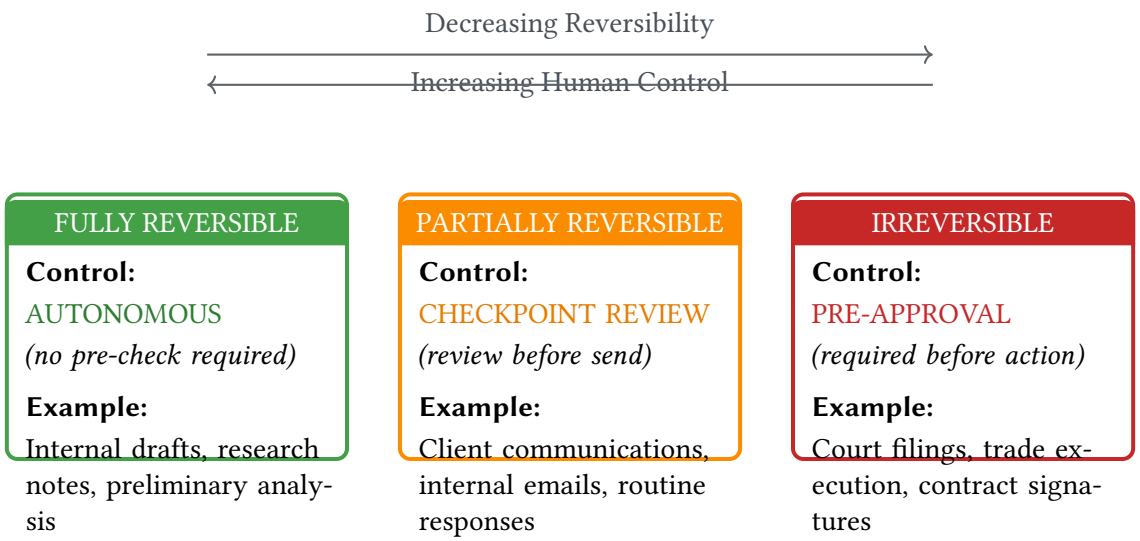

Figure 5: Oversight spectrum showing how the reversibility of agent actions determines the level of human control required. Fully reversible actions may proceed autonomously, partially reversible actions require checkpoint review, and irreversible actions demand pre-approval before execution.

9.4 Domain-Specific Escalation Requirements

Regulated industries impose specific escalation duties based on professional responsibility and compliance rules.

Legal Practice. Escalation requirements arise from professional responsibility rules (American Bar Association 2025; American Bar Association Standing Committee on Ethics and Professional Responsibility 2024):

- **Competence Limits:** Escalate when matters exceed agent training (ABA Model Rule 1.1).
- **Privilege Protection:** Escalate any action that might expose privileged information.
- **Conflicts of Interest:** Escalate potential conflicts to counsel.
- **Candor to Tribunal:** Escalate immediately if adverse authority requires disclosure.

Financial Services. Requirements arise from regulatory obligations and fiduciary duties (Financial Industry Regulatory Authority 2024; Board of Governors of the Federal Reserve System and Office of the Comptroller of the Currency 2011):

- **Suitability:** Escalate investment recommendations for adviser review.
- **Regulatory Thresholds:** Escalate when trades approach reporting limits.
- **MNPI:** Escalate immediately if potential Material Non-Public Information is encountered.
- **Risk Limits:** Escalate actions that would breach position limits.

9.5 Evaluating Escalation Mechanisms

Assess escalation mechanisms against six criteria:

Coverage: Do all appropriate situations trigger escalation? Test edge cases and conflicting data.

Calibration: Are thresholds set correctly? Avoid escalating everything (low value) or nothing (high risk).

Latency: Does escalation reach the human quickly enough? Urgent matters require immediate notification.

Routing: Does it reach the *right* person? Legal questions go to attorneys; risk breaches go to managers.

Context Quality: Can the human decide based on the message alone? If they must investigate from scratch, the handoff failed.

Response Handling: Does the agent correctly incorporate the human's guidance? Test the full cycle.

9.6 From Escalation to Delegation

Escalation moves control *up* (vertical) to a supervisor. Agents can also move control *sideways* (horizontal) by delegating to specialists.

Section 10 examines this next question: how does an agent work with other agents? Delegation patterns enable complex workflows where coordinating agents assign tasks to specialists (research,

drafting, modeling), while each retains the ability to escalate vertically when needed. Together, escalation and delegation define the full topology of human-agent collaboration.

10 How Does an Agent Work with Other Agents?

Complex matters require coordination. An M&A partner does not execute everything personally; they coordinate specialists. Corporate counsel reviews governance, tax specialists analyze structure, and antitrust counsel assesses regulatory risk. Each specialist has deep expertise in their domain. The partner orchestrates: defining deliverables, integrating work products, and synthesizing conclusions for the client.

A portfolio manager coordinates similarly: analysts provide company analysis, traders handle execution, risk managers monitor exposure, and compliance officers verify adherence. Complex trades require all these perspectives; no single person possesses all necessary expertise.

Agentic systems face the same coordination challenge. A single agent trying to do everything quickly exceeds its competence, permission boundaries, or context limits. Multi-agent architectures mirror professional teams: specialized agents with deep expertise, orchestrators that coordinate them, and protocols that enable collaboration.

Delegation

Delegation assigns subtasks from one agent (the coordinator) to another (the specialist). Unlike escalation (agent to human), delegation is agent to agent. The coordinator defines *what* needs to be done; the specialist determines *how*.
Delegation enables parallelization (multiple specialists work simultaneously), specialization (each agent is optimized for its domain), and security isolation (each agent has only the permissions it needs).

10.1 Why Multi-Agent Architectures?

Several factors drive multi-agent designs (Wu et al. 2023; Guo et al. 2024).

Specialization allows agents to excel in narrow domains. A securities law agent can be optimized for SEC regulations and equipped with EDGAR tools. A tax agent handles tax implications. Neither needs expertise in the other's domain.

Security isolation enforces least privilege. A research agent can read legal databases but cannot file documents. A filing agent can submit to CM/ECF but cannot access client financial data. If one agent is compromised, damage is contained.

Parallel execution lets independent workstreams proceed simultaneously. A document review

agent analyzes contracts while a research agent investigates legal issues. Neither waits for the other.

Vendor diversity enables best-of-breed selection. A specialized legal model handles research; a general model handles drafting; a fast model handles classification.

Scale management addresses context window limits. Rather than cramming everything into one context, you decompose the task across agents, each with focused context.

The tradeoffs are coordination overhead (communication costs), debugging complexity (failures span agents), and increased attack surface.

10.2 Agent-to-Agent Protocol (A2A)

The Agent-to-Agent Protocol (A2A) standardizes collaboration, complementing MCP (Google Developers 2025; Anthropic 2024). If MCP is how agents access resources, A2A is how agents delegate work.

A2A uses familiar professional concepts:

- **Agent Cards:** Capability statements (like a résumé) listing expertise, inputs, and outputs.
- **Tasks:** Units of delegated work (like engagement letters) specifying scope, constraints, and deadlines.
- **Artifacts:** Deliverables returned upon completion (memos, reports, data).
- **Channels:** Communication streams for status updates and clarification during execution.

A2A Task Delegation

The task lifecycle mirrors professional delegation:

1. **Discovery:** Finding the right specialist via Agent Card (like finding co-counsel).
2. **Delegation:** Creating a Task with goals and constraints (like an engagement letter).
3. **Execution:** The specialist works independently, requesting clarification if needed.
4. **Delivery:** The specialist returns Artifacts (draft memos).
5. **Completion:** The coordinator reviews and approves the work.

10.3 Multi-Agent Patterns

Three patterns organize collaboration (Wang et al. 2024b).

Sequential Delegation: Specialists work in series. The Coordinator delegates to a Research Agent; output flows to an Analysis Agent, then to a Drafting Agent. This is simple but slow.

Parallel Delegation: Specialists work simultaneously. Securities, Tax, and Employment Agents analyze an acquisition concurrently. The Coordinator integrates findings. This is faster but requires

task independence.

Hierarchical Delegation: Specialists delegate to sub-specialists. A Lead Due Diligence Agent delegates to Document Review and Legal Research agents. This handles complexity but introduces overhead.

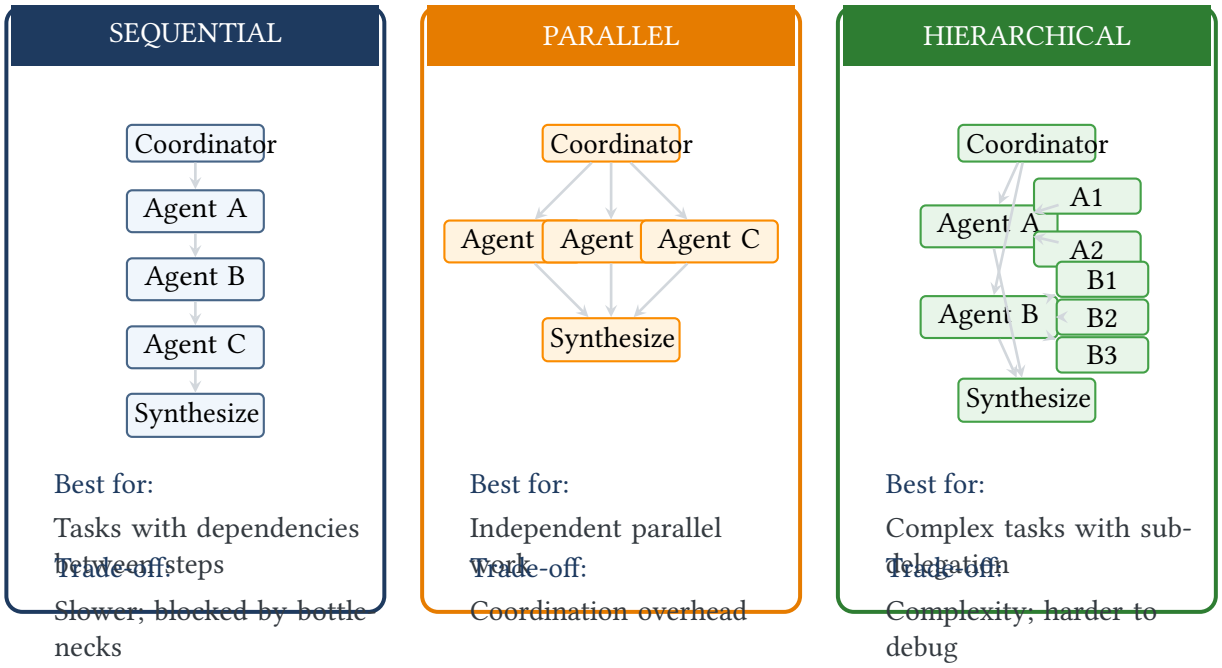

Figure 6: Three multi-agent orchestration patterns. Sequential delegation chains agents in order, ideal for dependent tasks but vulnerable to bottlenecks. Parallel delegation runs agents concurrently, maximizing throughput for independent work but requiring coordination. Hierarchical delegation enables sub-agents to handle specialized sub-tasks, providing flexibility for complex workflows at the cost of debugging complexity.

Production systems combine MCP and A2A. MCP handles agent-to-tool communication (database access). A2A handles agent-to-agent coordination (task delegation).

Consider M&A due diligence. The Orchestrator delegates via A2A to specialists. The Document Processing Agent uses MCP to access the data room. The Financial Analysis Agent uses MCP to query financial databases. Each returns Artifacts via A2A, which the Orchestrator synthesizes.

10.4 Multi-Agent Workflow Examples

Legal: Regulatory Assessment. A fintech company asks about regulatory approvals for a new product. The Orchestrator decomposes the request:

- **Securities Agent:** Analyzes SEC guidance via MCP; identifies potential registration requirements.
- **Banking Agent:** Checks OCC/FDIC rules via MCP; identifies money transmitter license needs.
- **Consumer Agent:** Reviews CFPB rules; flags disclosure requirements.

- **AML Agent:** Analyzes Bank Secrecy Act obligations; determines KYC needs.

The Orchestrator synthesizes these into a prioritized regulatory roadmap.

Finance: Block Trade Execution. A portfolio manager requests a $50M block trade. The Orchestrator decomposes the execution:

- **Market Agent:** Analyzes liquidity via MCP; recommends VWAP execution over two days.
- **Compliance Agent:** Verifies limits via MCP; notes 13F reporting requirement.
- **Risk Agent:** Assesses portfolio impact via MCP; confirms risk remains within policy.
- **Execution Agent:** Places orders via MCP order management system.

The Orchestrator monitors progress and reports completion.

10.5 Multi-Agent Risks

Multi-agent systems introduce specific failure modes.

Coordination Failures:

- **Deadlock:** Agents wait for each other cyclically. Prevention requires timeouts and dependency checks.
- **Divergence:** Specialists reach incompatible conclusions. The orchestrator must reconcile or escalate.
- **Cascading Errors:** Incorrect output propagates. Prevention requires input validation at each handoff.

Security Risks (OpenID Foundation AI Identity Management Community Group 2024):

- **Identity:** Agents must have verifiable, cryptographic identities to prevent impersonation.
- **Authorization:** Access policies must restrict which agents can delegate to whom.
- **Information Barriers:** Ethical walls must enforce conflict rules across agent boundaries.
- **Auditability:** You must log every delegation (who, what, when) for compliance.

10.6 Protocol Selection Guidance

Use MCP for tools. Use A2A for agents. Use both for complex workflows. As of late 2025, MCP is production-ready ("Model Context Protocol Specification" 2025). A2A is maturing but cross-vendor reliability varies ("Agent2Agent Protocol Specification" 2025). Design fallbacks to human coordination where necessary.

In practice, the boundary is this: use MCP when you need a well-defined, auditable operation ("fetch this filing," "calculate this metric"). Use A2A when you need a specialist to run a longer sub-process ("research and synthesize," "draft and revise," "coordinate across constraints") and you want structured status updates and artifacts.

Table 4: Protocol selection cues

Signal	Protocol	Latency	Examples
Immediate, well-defined operation	MCP	ms–seconds	Query database; retrieve document; run calculation
Delegated work requiring judgment	A2A	minutes–hours	Assign research; request analysis; coordinate specialists
End-to-end workflow with both	MCP + A2A	blended	Due diligence; portfolio rebalancing; regulatory assessment

10.7 From Delegation to Governance

Delegation distributes work, creating governance challenges. Accountability becomes complex: does responsibility lie with the coordinator, the specialist, or the approving human? Information barriers must apply to agents just as they do to humans. Audit trails must span the entire delegation tree.

Section 11 previews these requirements. Chapter 8 (Agents Part III: Governing Agents) develops them in detail, translating multi-agent delegation into concrete controls: delegation policy, identity and authorization, logging and retention, and organizational accountability.

11 How Do We Keep the Agent Safe?

Every professional organization relies on infrastructure to ensure safety. A law firm does not rely solely on an associate's ethics; it has conflicts committees, billing review systems, and opinion committees. A bank does not rely solely on a trader's judgment; it has risk management engines, compliance monitoring, and internal audit functions. These structures do not do the work; they ensure the work is done safely.

Agentic systems require the same infrastructure. Governance is not a single question but a lens through which all other questions must be viewed. Every capability creates a governance requirement. Every architectural choice enables or constrains oversight. This section synthesizes the governance implications of the ten questions.

Agentic System Governance

Agentic system governance encompasses the policies, controls, and oversight mechanisms that ensure agents operate safely, ethically, and in compliance with applicable requirements. Governance is not optional for regulated services (American Bar Association Standing Committee on Ethics and Professional Responsibility 2024; Financial Industry Regulatory Authority 2024). Professional duties are non-delegable: attorneys remain liable for AI-assisted work

product, and fiduciaries remain accountable for AI-informed recommendations.

11.1 Architecture Enables Governance

The architectural choices in this chapter are not merely technical decisions; they are the *infrastructure* that makes governance possible.

You cannot audit what you did not log. You cannot enforce privilege boundaries that were never implemented. You cannot demonstrate bounded operation without termination mechanisms. When a regulator asks how the compliance agent detected a breach, or when opposing counsel demands the agent's reasoning, architecture determines whether you can answer.

This chapter provided the technical architecture. The following chapter details the policy obligations. Section 13.1 maps these choices to their governance implications.

11.2 Governance Requirements by Question

Each of the ten questions creates specific governance requirements:

Table 5: Ten-question governance mapping

Q	Question	Governance Requirement
1	Triggers	Event authorization, audit logging of all triggers
2	Intent	Purpose limitation, goal alignment verification
3	Perception	Data governance, access controls, provenance tracking
4	Action	Actuation controls, approval gates, rollback capability
5	Memory	State integrity, retention policies, isolation enforcement
6	Planning	Bounded operation, resource budgets, plan validation
7	Termination	Exit protocols, success criteria, graceful degradation
8	Escalation	Human oversight, escalation triggers, response tracking
9	Delegation	Agent identity, delegation contracts, barrier enforcement
10	Governance	Meta-governance, audit architecture, compliance monitoring

11.3 Security Essentials

Five security controls are essential for regulated agentic systems (OWASP Foundation 2025; "Information technology — Artificial intelligence — Management system" 2023):

Security Controls for Regulated Practice

1. **Input Separation:** Isolate user inputs from system prompts to prevent prompt injection attacks.
2. **Output Validation:** Verify agent outputs before execution to detect hallucinations and policy violations.
3. **Least Privilege:** Grant minimum necessary tool access to limit the blast radius of failures.
4. **Audit Logging:** Maintain comprehensive action logs for accountability and investigation.
5. **Matter/Client Isolation:** Enforce confidentiality boundaries to protect privileged information.

These controls map directly to the framework: Input Separation protects Intent (Q2). Output Validation governs Action (Q4). Least Privilege limits Perception (Q3) and Action (Q4). Audit Logging enables review of Termination (Q7) and Escalation (Q8). Isolation enforces Memory (Q5) boundaries.

11.4 Transparency and Explainability

Regulators and clients require explanations for agent decisions (Zhong and Goel 2024). We define four transparency levels serving different audiences:

Level 0 (Output Only): The answer alone ("Suspicious transaction flagged"). Sufficient for low-stakes, high-trust contexts.

Level 1 (Summary + Sources): Conclusion with citations ("Flagged transaction #45921; exceeds Rule 203(b)(1)"). Enables verification.

Level 2 (Reasoning Outline): Key analytical steps ("Flagged because: (1) $150K > $100K threshold, (2) counterparty on watchlist"). Appropriate for substantive work product.

Level 3 (Execution Trace): Full structured record of tool calls, retrieval, and reasoning. Essential for audit and debugging.

The architecture must capture Level 3 traces for all operations, then generate audience-appropriate summaries (Levels 0–2) on demand.

11.5 Auditability vs. Retention

Tension exists between comprehensive logging (audit) and data minimization (privacy). The solution is not "log everything forever." We apply four practices:

Structured Logging captures decisions rather than raw chain-of-thought. This enables selective retention of decision points while discarding ephemeral reasoning.

Tiered Retention implements purpose-specific periods:

- **Short-term (days):** Full operational logs for debugging.
- **Medium-term (months):** Structured decision logs for audit.
- **Long-term (years):** Minimal compliance archives as required by regulation.

Redaction at Capture applies privacy filters before logging. Once sensitive data enters logs, systematic removal is difficult.

Legal Hold Integration ensures retention schedules yield to preservation obligations when litigation is anticipated.

11.6 Forward to Chapter 8

This chapter answered *how to build an agentic system.* The ten questions provided the architectural foundations; each created governance requirements the system must support.

The following chapter answers: *how do we govern these systems?* It examines the five-layer governance stack (legal, model, system, process, culture), dimensional controls, and compliance frameworks—including Federal Reserve guidance (Board of Governors of the Federal Reserve System and Office of the Comptroller of the Currency 2011) and the NIST AI RMF (National Institute of Standards and Technology 2023).

Architecture provides the foundation. Governance provides the controls. Together, they enable responsible deployment.

12 Synthesis: Reference Architectures

The previous sections examined each of the ten questions in isolation. This section shows how they work together in complete deployments. Two reference architectures demonstrate the full framework while honestly acknowledging current limitations: one legal, one financial.

Reference Architectures, Not Production Claims

These architectures illustrate how components fit together as *design targets*, not claims about current reliability. The reliability cliff (Section 8.5) constrains what agentic systems achieve today; these multi-hour workflows require extensive human oversight. Read as "how you would design it" not "how it works today."

12.1 Case Study: Credit Facility Documentation Review

A corporate client is borrowing $500 million under a senior secured revolving credit facility. The law firm represents the borrower. The partner assigns the matter: "Review the draft credit agreement and identify provisions that differ materially from market terms or our standard positions."

This is a document review task that would traditionally require 8–12 hours of senior associate time. The goal is not to replace the associate but to accelerate the review and ensure comprehensive coverage. The ten-question framework guides every design choice.

Q1 (Triggers): Work enters via document upload to the deal room and partner assignment. The trigger is explicit: new document plus assignment.

Q2 (Intent): The agent extracts intent: document review task, borrower perspective, focus on material deviations. Implicit constraints include confidentiality (privileged work product) and deadline (closing in two weeks).

Q3 (Perception): The agent uses MCP Resources to access the draft agreement (document management), the firm's template (precedent database), and market terms data (external database). Access is read-only.

Q4 (Action): Action tools are limited to document annotation (internal markup) and memo generation (work product creation). No external actions like filing or communication are permitted.

Q5 (Memory): Episodic memory tracks analysis progress (sections reviewed, issues identified). RAG provides access to prior credit agreement memos. Matter isolation ensures this work doesn't access unrelated client information.

Q6 (Planning): Plan-Execute pattern. The agent creates a section-by-section review plan based on the table of contents. It executes systematically through financial covenants, events of default, and representations.

Q7 (Termination): Success criteria: all material sections reviewed, issues identified, draft memo produced. Budget: token limit, time limit, iteration limit. Checkpoint after initial scan to confirm scope.

Q8 (Escalation): Escalate on ambiguous provisions requiring legal judgment, potential conflicts, or issues affecting deal viability. Use the Human-as-Tool pattern for partner input on materiality thresholds.

Q9 (Delegation): Single-agent architecture. Multi-agent would be appropriate if combined with separate research or financial modeling workstreams.

Q10 (Governance): Audit logging of all document access and analysis steps. Privilege protection enforced. Work product clearly marked as AI-assisted for attorney review.

The agent proceeds systematically:

1. Partner assigns matter; agent receives trigger.
2. Agent retrieves credit agreement, template, market terms.
3. Agent creates review plan: 15 sections.
4. Agent analyzes Section 1 (Definitions): identifies unusual defined terms.
5. Agent continues through financial covenants: flags leverage ratio differing from market.
6. ... [continues through all sections]
7. Agent compiles findings into issues list and draft memo.
8. Agent presents to associate for review.
9. Associate reviews, adds context, and escalates significant issues to partner.

Even well-designed agents fail. The agent may miss nuanced definitions where a term has been subtly modified. Human review catches these subtleties. Cross-document dependencies present another risk; if the agent does not analyze referenced schedules, issues may be missed. The most dangerous failures are omissions. Mitigation requires checkpoint review and partner validation.

12.2 Case Study: Equity Portfolio Management

An investment adviser manages a $200 million equity portfolio for institutional clients. The portfolio manager (PM) wants continuous monitoring with agent assistance for rebalancing analysis, compliance checking, and research synthesis.

This is a continuous monitoring task, not a one-time analysis. The goal is augmentation, not autonomous trading. The ten-question framework shapes this multi-agent architecture.

Q1 (Triggers): Multiple triggers: market data feeds (price movements), scheduled jobs (daily compliance check), human prompts (PM requests), and escalation events (limit breaches).

Q2 (Intent): Varies by trigger. Price alert: assess significance. Scheduled rebalance: generate trade list if needed. PM query: answer specific question.

Q3 (Perception): MCP Resources access market data, portfolio data, research, and compliance data. Read-only access to trading systems.

Q4 (Action): The agent generates recommendations and reports. Trade execution requires PM approval (approval gate). No autonomous trading.

Q5 (Memory): Episodic memory tracks recent analysis and PM decisions. RAG accesses research archive. Client isolation ensures segregation.

Q6 (Planning): ReAct pattern for ad hoc analysis. Plan-Execute for scheduled tasks. Hierarchical for comprehensive reviews.

Q7 (Termination): Monitoring is continuous. Analysis completes when the question is answered. Rebalancing completes when the trade list is generated and approved.

Q8 (Escalation): Escalate on positions approaching limits, unusual market conditions, conflicting

signals, or any trade recommendation.

Q9 (Delegation): Multi-agent architecture. Monitoring Agent watches market data. Research Agent synthesizes reports. Compliance Agent checks guidelines. Rebalancing Agent generates recommendations. PM Agent orchestrates.

Q10 (Governance): Comprehensive audit trail. Fiduciary duty documentation. Compliance monitoring. MNPI controls.

Multiple agents coordinate:

1. Monitoring Agent detects: "Tech sector up 3%; allocation 35% vs 25% target."
2. Monitoring Agent triggers Rebalancing Agent.
3. Rebalancing Agent queries Compliance Agent: "Check proposed sales against guidelines."
4. Compliance Agent confirms: trades within limits.
5. Rebalancing Agent queries Research Agent: "Any negative research?"
6. Research Agent returns: "One position has earnings next week."
7. Rebalancing Agent adjusts: defer that sale.
8. Rebalancing Agent presents recommendation to PM Agent.
9. PM Agent formats for human review.
10. Human PM reviews, approves, and authorizes execution.
11. Execution Agent places orders via OMS.

Coordination introduces failure vectors. Cascading errors occur if the Monitoring Agent misinterprets data. Debugging complexity increases with multiple agents. Mitigation requires comprehensive logging and clear escalation protocols.

12.3 Synthesis: Principles Across Domains

Both case studies illustrate common principles:

Decomposition is essential. Neither workflow attempts end-to-end autonomous completion. Tasks are broken into manageable steps with human checkpoints.

Human-in-the-loop oversight is the norm. Attorney review and PM approval are structural requirements. Agents augment professional judgment; they do not replace it.

Memory enables continuity. Both rely on episodic memory and RAG. Without memory, every interaction starts fresh.

Isolation is non-negotiable. Matter isolation (legal) and client isolation (financial) are architectural requirements.

Failure modes are predictable. Nuanced judgment failures, omissions, and cascading errors appear in both domains. Design for these failures explicitly.

Governance is pervasive. Every architectural choice has governance implications. Audit trails,

approval gates, and escalation paths are foundational.

Expectation setting is part of the work. In regulated professional services, it is not enough for an agent to be helpful; stakeholders must understand what it did, what it did not do, and what remains uncertain. Treat this as a deliverable requirement: outputs should clearly state scope, as-of dates, sources accessed, and any confidence caveats. Where outputs may be relied on externally (clients, counterparties, regulators), require explicit human review and use the system's logs to support disclosure and defensible documentation.

The reliability cliff remains binding. Even well-designed systems degrade sharply as tasks extend from minutes to hours (Section 8.5). Architect for short, check-pointed work with graceful partial completion and escalation, not for uninterrupted autonomy.

Chapter 8 (Agents Part III: Governing Agents) takes these themes from principles to controls: how to calibrate control intensity, implement audit and retention, and define accountability and disclosure practices that match professional obligations.

12.4 Framework Completion Checklist

Before deploying any agentic system, verify that all ten questions have been answered:

Ten-Question Deployment Checklist

- ☐ **Triggers:** How does work enter? Are all trigger types covered? Is there audit logging?
- ☐ **Intent:** How is intent extracted? What happens with ambiguity? Are constraints identified?
- ☐ **Perception:** What tools provide information? Is access controlled? Is provenance tracked?
- ☐ **Action:** What can the agent do? Are irreversible actions gated? Is rollback possible?
- ☐ **Memory:** What persists? Is isolation enforced? What are retention policies?
- ☐ **Planning:** What pattern applies? Are budgets enforced? Is there loop detection?
- ☐ **Termination:** How does it know when it's done? What are success criteria?
- ☐ **Escalation:** When does it ask for help? Who receives escalations? Is context sufficient?
- ☐ **Delegation:** Does it coordinate? Are protocols standardized? Are barriers enforced?
- ☐ **Governance:** Are security controls implemented? Is there audit capability?

Any question left unanswered represents a gap in the architecture that will manifest as a failure in production.

13 Conclusion: From Architecture to Governance

This chapter opened with a claim: **agents are not magic; they are architecture**. Ten sections later, that claim should feel concrete.

You now know how work reaches an agent (triggers), how instructions become goals (intent), how agents gather information (perception) and take action (action), how context persists (memory), how complex work decomposes (planning), how agents recognize completion (termination), when they hand off to humans (escalation), how they coordinate (delegation), and what controls keep them safe (governance).

Each capability involves tradeoffs. Richer memory improves context but increases latency. Aggressive escalation improves safety but reduces autonomy. Tighter approval gates reduce risk but slow execution. There are no free lunches, only choices that must be calibrated to your context, risk tolerance, and professional obligations. Chapter 8 (Agents Part III: Governing Agents) provides a risk-based calibration approach rather than a one-size-fits-all checklist.

The organizational analogy is not merely pedagogical. Law firms and investment teams are cognitive work systems that have evolved infrastructure for exactly these challenges: distributing work, maintaining context, and ensuring quality. When you evaluate an agentic system, ask the same questions you would ask about a professional team.

13.1 What This Understanding Enables

With architectural literacy, you can evaluate vendor claims. When a vendor says their agent "handles legal research," ask: What triggers it? How does it understand the question? What databases does it access? How does it know when it is done? What happens when confidence is low? The ten questions provide your evaluation framework.

You can participate meaningfully in procurement. Assess whether a system meets requirements: Does it enforce matter isolation? Maintain audit trails? Integrate with approval workflows? Escalate appropriately? You have the vocabulary to specify requirements.

You can demand governance artifacts, not promises. Ask vendors to demonstrate action gating, escalation behavior under low confidence, and reconstruction via logs. If a system cannot show you what it accessed, what it did, and why it stopped, it is not deployable in regulated practice.

You can design governance that maps to architecture. Governance is enabled by architecture. If you want audit trails, the system must log reasoning. If you want approval gates, the system must pause before action. If you want confidentiality, the system must isolate context. You can design systems where governance is built in, not bolted on.

Finally, you can communicate with technical teams. Describe requirements precisely: "I need perception tools for these databases, action tools behind approval gates, escalation triggers for low confidence, and memory isolation between matters." Shared vocabulary enables collaboration.

13.2 Honest Assessment of Current Capabilities

Architectural understanding requires honest acknowledgment of limitations.

The reliability cliff (Section 8.5) is the most significant constraint. Agents exhibit near-perfect success on short tasks but fail frequently on multi-hour workflows. Design systems that decompose work aggressively and checkpoint progress frequently.

Judgment limitations constrain value. Agents excel at retrieval and pattern matching but struggle with nuance and novelty. The effective deployments pair agent capabilities with human judgment.

Brittleness causes failures due to API changes or edge cases. Build monitoring and graceful degradation into every deployment.

Compounding errors affect multi-step workflows. Error probabilities multiply, so long autonomous chains fail. Shorter workflows with human checkpoints perform better.

13.3 Essential Resources

Four resource categories guide practitioners from concept to deployment.

Security fundamentals must be addressed before deployment. Implement the five controls from Section 11.3: input separation, output validation, least privilege, audit logging, and isolation. The OWASP LLM Top 10 provides vulnerability taxonomy.

Protocols and standards define interoperability. The Model Context Protocol (MCP) standardizes tool communication. The Agent-to-Agent Protocol (A2A) standardizes coordination.

Research foundations provide theoretical grounding. Key works include Xi et al. on architecture (Xi et al. 2023), Yao et al. on ReAct (Yao et al. 2022), and Park et al. on memory (Park et al. 2023).

Regulatory guidance varies by domain. Legal practitioners should monitor ABA ethics opinions (especially 512). Financial practitioners should monitor SEC, FINRA, and Federal Reserve guidance.

Temporal Warning

Resources accurate as of late 2025 may not reflect subsequent developments. Protocols evolve, regulations change, and vulnerabilities emerge. Verify currency before relying on any reference.

13.4 From Architecture to Governance

This chapter answered: *How do you build an agent?* The next chapter answers: *How do you govern one?*

These questions are connected. Every architectural decision has governance implications:

- Trigger logging creates audit trails.

- Intent extraction allows review of understanding.
- Perception controls enforce data governance.
- Action gates require approval workflows.
- Memory isolation protects privilege.
- Planning budgets ensure bounded operation.
- Termination criteria verify completion.
- Escalation paths route uncertainty to humans.
- Delegation contracts establish accountability.

Architecture enables governance. If you did not architect for logging, you cannot audit. If you did not architect for isolation, you cannot enforce boundaries. Governance emerges from design decisions.

The governance imperative is that agents are not just tools. They interpret goals, select what to perceive, and take actions that can create liability. That shift introduces predictable accountability problems: purpose drift (misaligned goals), perceptual opacity (bad or manipulated inputs), and actuation risk (high-consequence actions). Chapter 8 builds on this foundation, translating these challenges into concrete controls: calibrated autonomy, input and action constraints, auditability, retention, escalation and override mechanisms, and accountability structures.

You cannot govern what you do not understand.

You cannot control what you did not architect.

Now you can do both.

References

Administrative Office of the U.S. Courts (2024a). *Electronic Filing (CM/ECF)*. Case Management/Electronic Case Files system for filing federal court documents online; mandatory for most federal court filings. URL: https://www.uscourts.gov/court-records/electronic-filing-cm-ecf (visited on 12/13/2025).

Administrative Office of the U.S. Courts (2024b). *Public Access to Court Electronic Records (PACER)*. Official PACER system providing electronic public access to federal appellate, district, and bankruptcy court records. URL: https://pacer.uscourts.gov/ (visited on 12/13/2025).

Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents (2024). Benchmark framework for agent security across 10 scenarios, 400+ tools, 27 attack/defense methods; includes prompt injection, memory poisoning, and backdoor attacks. URL: https://openreview.net/forum?id=V4y0CpX4hK (visited on 12/13/2025).

Agent2Agent Protocol Specification (2025). Official specification for A2A protocol; supports HTTP, SSE, gRPC transports; security card signing added November 2025. URL: https://a2a-protocol.org/latest/ (visited on 11/27/2025).

American Bar Association (2025). *Model Rules of Professional Conduct, Rule 1.1: Competence*. Primary authority establishing duty of competence including technology competence (Comment 8 amended 2012). URL: https://www.americanbar.org/groups/professional_responsibility/publications/model_rules_of_professional_conduct/rule_1_1_competence/ (visited on 12/13/2025).

American Bar Association Standing Committee on Ethics and Professional Responsibility (July 2024). *Formal Opinion 512: Generative Artificial Intelligence Tools*. Tech. rep. Addresses ethical obligations when using generative AI; covers competence, confidentiality, supervision, and billing. American Bar Association. URL: https://www.americanbar.org/groups/professional_responsibility/publications/ethics_opinions/formal-opinion-512/ (visited on 11/27/2025).

Anthropic (Nov. 2024). *Introducing the Model Context Protocol*. Open standard for connecting AI systems to data sources; pre-built servers for Google Drive, Slack, GitHub, Postgres; SDKs for Python, TypeScript, C#. URL: https://www.anthropic.com/news/model-context-protocol (visited on 11/27/2025).

Board of Governors of the Federal Reserve System and Office of the Comptroller of the Currency (Apr. 2011). *Supervisory Guidance on Model Risk Management*. Tech. rep. SR Letter 11-7 / OCC Bulletin 2011-12. Foundational guidance on model risk management for financial institutions; defines model risk, validation requirements, and governance standards; increasingly relevant to AI/ML models. Federal Reserve Board. URL: https://www.federalreserve.gov/supervisionreg/srletters/sr1107.htm (visited on 12/13/2025).

Bommarito, Michael James, Jillian Bommarito, and Daniel Martin Katz (Nov. 2025). *What is an Agent? A Conceptual Primer and History of Agents and Agentic AI*. Working Paper 5806982. Part I of the Agents series from *Artificial Intelligence for Law and Finance*; establishes GPA+IAT framework synthesizing seven decades of agency scholarship from philosophy, psychology, law, economics, and computer science. SSRN. DOI: 10.2139/ssrn.5806982. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5806982 (visited on 12/14/2025).

Dahl, Matthew, Varun Magesh, Mirac Suzgun, and Daniel E. Ho (2024). "Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools". In: *Journal of Empirical Legal Studies*. Finds RAG-based legal tools (Lexis+ AI, Westlaw AI) hallucinate 17–33% of the time; challenges vendor claims of hallucination-free performance. DOI: 10.1111/jels.12394. URL: https://law.stanford.edu/wp-content/uploads/2024/05/Legal_RAG_Hallucinations.pdf (visited on 12/13/2025).

"Evaluating Human-AI Collaboration: A Review and Methodological Framework" (2024). In: *arXiv preprint arXiv:2407.19098*. Framework for understanding approval gates and human oversight in AI systems; addresses goals, interaction, and task allocation factors. URL: https://arxiv.org/abs/2407.19098 (visited on 12/13/2025).

Financial Industry Regulatory Authority (June 2024). *Regulatory Notice 24-09: FINRA Reminds Member Firms of Regulatory Obligations When Using Artificial Intelligence*. Tech. rep. Official guidance establishing supervision and governance requirements for AI in financial services under Rule 3110. FINRA. URL: https://www.finra.org/rules-guidance/notices/24-09 (visited on 12/13/2025).

Free Law Project (Feb. 2023). *Launching Webhooks to Finally Automate Legal Data*. CourtListener webhook API for real-time PACER docket updates, search alerts, and RECAP fetch notifications; enables event-driven legal monitoring workflows. URL: https://free.law/2023/02/16/webhooks-for-legal-data (visited on 12/04/2025).

Google Developers (Apr. 2025). *Announcing the Agent2Agent Protocol (A2A)*. Open protocol for agent-to-agent communication using JSON-RPC 2.0 over HTTP; donated to Linux Foundation; features Agent Cards for capability discovery; 50+ launch partners. URL: https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/ (visited on 11/27/2025).

Guo, Taicheng, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang (2024). "Large Language Model Based Multi-agents: A Survey of Progress and Challenges". In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)*. Comprehensive survey on LLM-based multi-agent systems covering architectures, coordination mechanisms, and problem-solving capabilities. URL: https://arxiv.org/abs/2402.01680 (visited on 12/13/2025).

"Human-AI collaboration is not very collaborative yet: a taxonomy of interaction patterns in AI-assisted decision making from a systematic review" (2024). In: *Frontiers in Computer Science*. Systematic review of 105 articles developing taxonomy of human-AI interaction patterns and

handoff protocols. DOI: 10.3389/fcomp.2024.1521066. URL: https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2024.1521066/full (visited on 12/13/2025).

"Human-in-the-loop machine learning: a state of the art" (2022). In: *Artificial Intelligence Review*. Survey covering confidence-based escalation approaches: least confidence, margin of confidence, ratio of confidence, and entropy-based methods. DOI: 10.1007/s10462-022-10246-w. URL: https://link.springer.com/article/10.1007/s10462-022-10246-w (visited on 12/13/2025).

Information technology — Artificial intelligence — Management system (Dec. 2023). Standard. First international standard for AI management systems (AIMS); provides structured approach for governance using PDCA methodology. International Organization for Standardization. URL: https://www.iso.org/standard/81230.html (visited on 12/13/2025).

Kadavath, Saurav, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, et al. (2022). *Language Models (Mostly) Know What They Know*. Anthropic study on LLM confidence calibration; finds models can predict their own accuracy but require careful prompting; relevant to confidence thresholds for agent escalation. arXiv: 2207.05221 [cs.CL]. URL: https://arxiv.org/abs/2207.05221 (visited on 12/13/2025).

Kim, Sehoon, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami (2024). "An LLM Compiler for Parallel Function Calling". In: *International Conference on Machine Learning (ICML)*. Compiler-inspired architecture for parallel function execution in agents; demonstrates up to 3.7x latency speedup and 6.7x cost savings vs. ReAct. URL: https://arxiv.org/abs/2312.04511 (visited on 12/13/2025).

"Learning to Undo: Rollback-Augmented Reinforcement Learning with Reversibility Signals" (2024). In: *arXiv preprint arXiv:2510.14503*. Framework for reversibility and rollback in safety-sensitive AI systems; achieved 99.8% reduction in catastrophic failures through selective state rollback. URL: https://arxiv.org/abs/2510.14503 (visited on 12/13/2025).

Legal Information Institute (2024). *Rule 12. Defenses and Objections: When and How Presented*. Federal rule establishing 21-day deadline for responsive pleadings; critical for deadline calculation in litigation agents. URL: https://www.law.cornell.edu/rules/frcp/rule_12 (visited on 12/13/2025).

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. Original RAG paper introducing the paradigm of combining parametric (LLM) and non-parametric (retrieval) memory for generation tasks, pp. 9459–9474. URL: https://arxiv.org/abs/2005.11401 (visited on 12/13/2025).

METR (Mar. 2025). *Measuring AI Ability to Complete Long Tasks*. Empirical study finding AI agent success rates inversely correlated to task duration; 100% success on tasks under 4 minutes, under

10% for tasks over 4 hours; capability doubling time approximately 7 months. URL: https://metr.org/blog/2025-03-19-measuring-ai-ability-to-complete-long-tasks/ (visited on 11/27/2025).

Model Context Protocol Specification (2025). Official MCP documentation and specification; server and client SDKs; community server directory. URL: https://modelcontextprotocol.io/ (visited on 11/27/2025).

National Institute of Standards and Technology (Jan. 2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. Tech. rep. NIST AI 100-1. Voluntary framework for managing AI risks; organized around governance, mapping, measuring, and managing AI risks; referenced by EU AI Act and other regulatory frameworks. U.S. Department of Commerce. URL: https://www.nist.gov/itl/ai-risk-management-framework (visited on 12/13/2025).

Nielsen, Jakob (June 2023). *AI: First New UI Paradigm in 60 Years*. Introduces concept of intent-based outcome specification as the third UI paradigm after batch processing and command-based interaction; argues AI shifts locus of control from user specifying commands to user specifying desired outcomes. URL: https://www.nngroup.com/articles/ai-paradigm/ (visited on 12/13/2025).

"On the Impact of Event-Driven Architecture on Performance: An Exploratory Study" (2023). In: *Future Generation Computer Systems* 153. Comparative study evaluating event-driven vs. monolithic architecture on CPU, RAM, response time, and throughput metrics, pp. 1–15. DOI: 10.1016/j.future.2023.10.021. URL: https://dl.acm.org/doi/10.1016/j.future.2023.10.021 (visited on 12/13/2025).

OpenID Foundation AI Identity Management Community Group (2024). "Identity Management for Agentic AI: The New Frontier of Authorization, Authentication, and Security for an AI Agent World". In: *arXiv preprint arXiv:2510.25819*. Addresses authentication, authorization, and identity management in multi-agent systems; proposes DIDs, Verifiable Credentials, and Zero Trust principles. URL: https://arxiv.org/abs/2510.25819 (visited on 12/13/2025).

OWASP Foundation (2025). *OWASP Top 10 for Large Language Model Applications*. Security vulnerabilities in LLM applications; ranks prompt injection as critical vulnerability #1. URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/ (visited on 11/27/2025).

Park, Joon Sung, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein (2023). "Generative Agents: Interactive Simulacra of Human Behavior". In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*. Introduces memory stream architecture with reflection for long-term agent behavior; foundational for episodic memory and learning in agent systems. ACM. DOI: 10.1145/3586183.3606763.

Rao, Anand S. and Michael P. Georgeff (1995). "BDI Agents: From Theory to Practice". In: *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*. Foundational work on Belief-Desire-Intention cognitive architecture for agents. AAAI Press, pp. 312–319. URL: https://cdn.aaai.org/ICMAS/1995/ICMAS95-042.pdf (visited on 12/13/2025).

Reimers, Nils and Iryna Gurevych (2019). “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Foundational work on sentence embeddings for semantic similarity and retrieval; basis for modern vector search in RAG systems. Association for Computational Linguistics. URL: https://arxiv.org/abs/1908.10084 (visited on 12/13/2025).

Schick, Timo, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom (2023). “Toolformer: Language Models Can Teach Themselves to Use Tools”. In: *arXiv preprint arXiv:2302.04761*. Demonstrates LLMs learning to use external tools.

U.S. Securities and Exchange Commission (2024). *EDGAR Application Programming Interfaces (APIs)*. Official RESTful APIs for EDGAR submissions and XBRL data; real-time updates with sub-second processing delay; no authentication required for public APIs. URL: https://www.sec.gov/search-filings/edgar-application-programming-interfaces (visited on 12/13/2025).

“Using AI Uncertainty Quantification to Improve Human Decision-Making” (2023). In: *arXiv preprint arXiv:2309.10852*. Research on how uncertainty estimates guide autonomous vs. human-deferral decisions; addresses confidence calibration challenges. URL: https://arxiv.org/abs/2309.10852 (visited on 12/13/2025).

“Using Event Sourcing and CQRS to Build a High Performance Point Trading System” (2019). In: *Proceedings of the 2019 5th International Conference on E-Business and Applications*. Demonstrates CQRS and event sourcing with Actor model for high-performance systems; empirical evidence of scalability gains. ACM. DOI: 10.1145/3317614.3317632. URL: https://dl.acm.org/doi/10.1145/3317614.3317632 (visited on 12/13/2025).

Wang, Lei, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen (2023). “A Survey on Large Language Model based Autonomous Agents”. In: *arXiv preprint arXiv:2308.11432*. Comprehensive survey of LLM-based agents covering profile, memory, planning, and action modules with unified framework. URL: https://arxiv.org/abs/2308.11432 (visited on 12/13/2025).

Wang, Qingyun et al. (2025). “LiveMCPBench: Can Agents Navigate an Ocean of MCP Tools?” In: *arXiv preprint arXiv:2508.01780*. Benchmark evaluating agent performance over large MCP tool ecosystems; discusses MCP server ecosystem scale (10,000+ servers). DOI: 10.48550/arXiv.2508.01780. URL: https://arxiv.org/abs/2508.01780 (visited on 12/15/2025).

Wang, Wenxuan, Juluan Shi, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael R. Lyu (2024a). *Learning to Ask: When LLM Agents Meet Unclear Instruction*. Proposes Ask-when-Needed (AwN) framework prompting LLMs to ask questions when encountering unclear instructions; creates NoisyToolBench benchmark for imperfect instructions. arXiv: 2409.00557 [cs.CL]. URL: https://arxiv.org/abs/2409.00557 (visited on 12/13/2025).

Wang, Yaoxiang, Zhiyong Wu, Junfeng Yao, and Jinsong Su (2024b). “TDAG: A Multi-Agent Framework based on Dynamic Task Decomposition and Agent Generation”. In: *Neural Networks*. Addresses dynamic task decomposition in multi-agent systems; demonstrates complex task breakdown with specialized subagents. URL: https://arxiv.org/abs/2402.10178 (visited on 12/13/2025).

Wu, Qingyun, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang (2023). “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation”. In: *arXiv preprint arXiv:2308.08155*. Foundational framework for multi-agent conversation systems; covers customizable agents, flexible conversation patterns. URL: https://arxiv.org/abs/2308.08155 (visited on 12/13/2025).

Xi, Zhiheng, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. (2023). “The Rise and Potential of Large Language Model Based Agents: A Survey”. In: *arXiv preprint arXiv:2309.07864*. Comprehensive survey of LLM-based agents.

Xu, Binfeng, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu (2023). *ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models*. Modular paradigm separating reasoning from tool observations; achieves 5x token efficiency and 4% accuracy improvement over ReAct on HotpotQA. arXiv: 2305.18323 [cs.CL]. URL: https://arxiv.org/abs/2305.18323 (visited on 12/13/2025).

Yao, Shunyu, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao (2022). “ReAct: Synergizing Reasoning and Acting in Language Models”. In: *arXiv preprint arXiv:2210.03629*. Introduces ReAct pattern: alternating reasoning and acting in language models.

Zhang, Michael J. Q., W. Bradley Knox, and Eunsol Choi (2024). *Modeling Future Conversation Turns to Teach LLMs to Ask Clarifying Questions*. Finds LLMs often respond by presupposing a single interpretation of ambiguous requests rather than asking clarifying questions; proposes preference labeling using simulated future turns to teach when clarification is needed. arXiv: 2410.13788 [cs.CL]. URL: https://arxiv.org/abs/2410.13788 (visited on 12/13/2025).

Zhong, Chen and Sunita Goel (2024). “Transparent AI in Auditing through Explainable AI”. In: *Current Issues in Auditing* 18.2. Focuses on explainable AI as mechanism for transparency and trustworthiness in audit applications, A1–A14. URL: https://publications.aaahq.org/cia/article/18/2/A1/12271/Transparent-AI-in-Auditing-through-Explainable-AI (visited on 12/13/2025).