Agents

Part II: How to Build an Agent

Architectures, Protocols, and Technical Evaluation

Michael J Bommarito II · Daniel Martin Katz · Jillian Bommarito

December 11, 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

This chapter answers a practical question: *How do you build an agentic system?*

The previous chapter established what agentic systems are through the GPA+IAT framework (Goals, Perception, Action plus Iteration, Adaptation, Termination). This chapter shows how to construct them, translating the six theoretical properties into concrete architectural choices: planning mechanisms, tool integrations, memory systems, escalation protocols, and governance controls. The treatment is conceptual rather than code-focused. You will understand how production systems are architected without needing to implement them yourself.

We organize the chapter around ten fundamental questions that every agentic system must answer. These questions map directly to organizational analogies: how work arrives (inbox and calendar), how instructions are understood (assignment memos), how information is gathered (library access), how actions are taken (filing and execution), how context is preserved (case files), how complex work is decomposed (project plans), how completion is recognized (deliverable criteria), when to escalate (going to the supervisor), how specialists coordinate (co-counsel relationships), and how safety is ensured (compliance and audit). Table 1 provides the complete mapping.

The chapter supports multiple reading strategies depending on your goals. Sequential readers should begin with the organizational analogy and ten-question framework below, then proceed through Questions 1–10 in order before concluding with the synthesis in Section 12. Random-access readers can jump directly to any question section, as each stands alone with self-contained explanations. Practitioners evaluating vendors should focus on questions most relevant to their procurement criteria; Section 12 provides complete reference architectures with failure mode analysis that maps directly to vendor evaluation checklists.

1.1 Agentic Systems as Organizations

An agentic system functions like a professional organization. This observation serves as more than analogy; it is a design principle. Consider how a law firm operates: it has a mission (goals), gathers intelligence through research and client intake (perception), executes work through filing and communication (action), operates in project cycles and matter lifecycles (iteration), learns from experience and adjusts strategy (adaptation), and completes engagements or escalates beyond authority (termination). The six properties that define agentic systems map directly to how professional organizations function.

A discretionary portfolio management team follows the same pattern. A mandate or prospectus sets risk/return objectives (goals); the team watches market data, research, and issuer filings (perception); it rebalances portfolios and places orders with compliance checks (action); it works in daily and quarterly review cycles (iteration); it shifts allocations after performance and risk reviews (adaptation);

and it closes or cut positions, escalating to an investment committee when limits or mandates are at risk (termination). The organizational logic is identical even though the domain differs.

This mapping has practical implications. When we design an agentic system, we face the same questions that arise when designing an organization. The system requires specific capabilities, work must be divided among specialists, escalation paths must exist for problems exceeding authority, institutional knowledge must be preserved across matters, and governance controls must ensure safe operation. A partner staffing a complex transaction and an architect designing an agent system are solving structurally similar problems.

Table 1 makes this correspondence explicit, mapping each GPA+IAT property to the operational questions it generates. The sections that follow address these questions in order, using organizational analogies throughout to ground abstract concepts in familiar professional practice.

Table 1: From GPA+IAT properties to operational questions

Section	**GPA+IAT Property**	**Operational Question**
Triggers	Perception	How does it know when it has work?
Intent	Goal	How does it understand what's being asked?
Perception	Perception	How does it find things out?
Action	Action	How does it make things happen?
Memory	Adaptation	How does it remember things?
Planning	Goal + Iteration	How does it break work into steps?
Termination	Termination	How does it know when it's done?
Escalation	Termination	When does it ask for help?
Delegation	Iteration	How does it work with other agents?
Governance	All	How do we keep it safe?

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.

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).

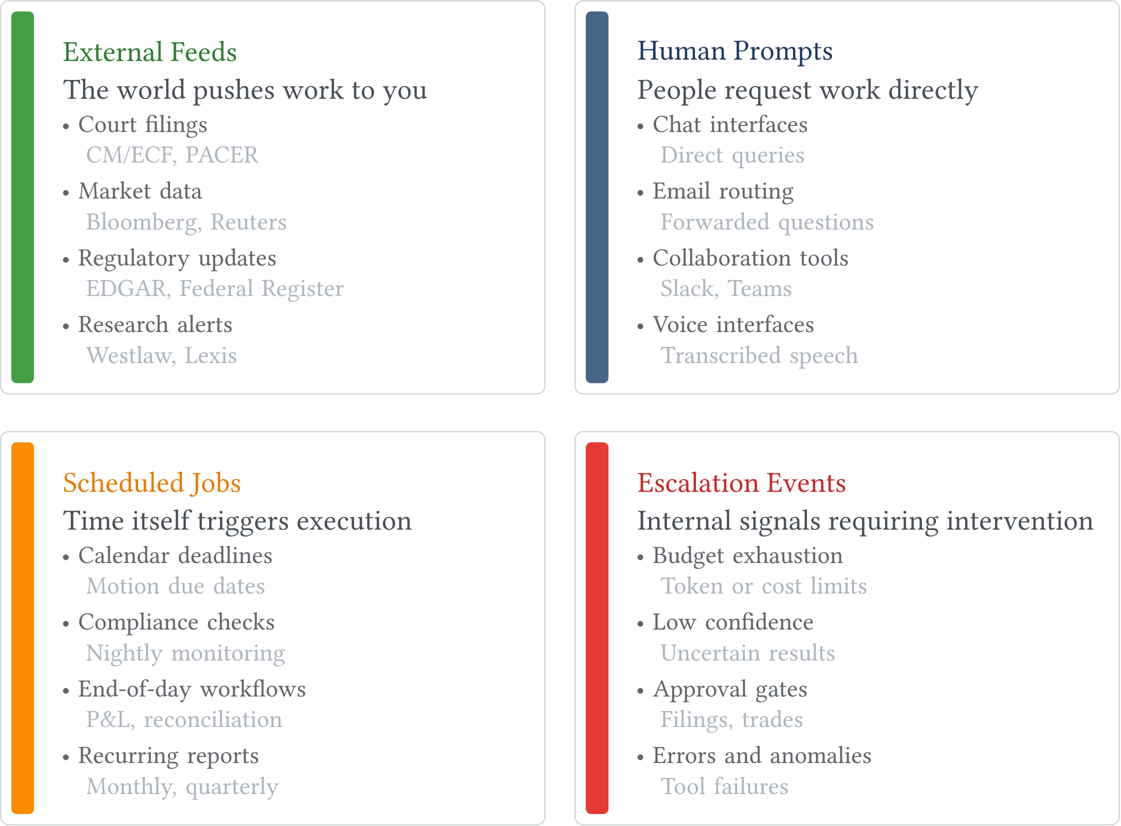
Before an agent can reason or act, it must first notice that work exists. Channels are the sensory apparatus of the system: the ways it becomes aware of its environment and the tasks it must accomplish—just as a lawyer cannot respond to a motion they never received.

2.1 External Feeds: The World Pushes Work to You

External feeds deliver events from systems outside the agent'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 (CM/ECF, state e-filing) send notifications when documents are filed; an agent receives these alerts, retrieves filed documents via PACER, analyzes contents, and triggers appropriate responses. The SEC's EDGAR system publishes corporate filings with programmatic access, enabling agents to monitor competitors' 10-Ks and flag material differences from your company's disclosures. Regulatory agencies publish through the Federal Register and agency websites. Citator alerts from Westlaw and Lexis notify when monitored cases are cited or overruled.

Financial Market Feeds: Financial institutions receive real-time market data through Bloomberg, Reuters, and similar providers. A portfolio management agent subscribes to price alerts, receives notifications when thresholds are crossed, evaluates rebalancing rules, and either executes trades within risk limits or escalates to a portfolio manager. Position and P&L updates cascade through financial systems: trades trigger position updates, which trigger risk recalculation, compliance checks, and dashboard updates. News feeds deliver headlines, earnings, and sentiment analytics; agents assess materiality and alert managers when news appears significant.



External Feeds

The world pushes work to you

- Court filings
  CM/ECF, PACER
- Market data
  Bloomberg, Reuters
- Regulatory updates
  EDGAR, Federal Register
- Research alerts
  Westlaw, Lexis

Human Prompts

People request work directly

- Chat interfaces
  Direct queries
- Email routing
  Forwarded questions
- Collaboration tools
  Slack, Teams
- Voice interfaces
  Transcribed speech

Scheduled Jobs

Time itself triggers execution

- Calendar deadlines
  Motion due dates
- Compliance checks
  Nightly monitoring
- End-of-day workflows
  P&L, reconciliation
- Recurring reports
  Monthly, quarterly

Escalation Events

Internal signals requiring intervention

- Budget exhaustion
  Token or cost limits
- Low confidence
  Uncertain results
- Approval gates
  Filings, trades
- Errors and anomalies
  Tool failures

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 agent system's event router.

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 agents add value in financial workflows.

Agents are not suited for: High-frequency trading, market-making, latency-sensitive execution. These domains require deterministic algorithms operating at microsecond latencies. An LLM reasoning loop, even a fast one, cannot compete.

Agents are suited for: Strategic portfolio decisions, investment thesis development, rebalancing analysis, compliance monitoring, research synthesis. These tasks operate on timescales of minutes to hours, where reasoning quality matters more than latency.

The architecture pattern: Fast deterministic systems handle real-time data capture and threshold detection. When thresholds trigger (position approaching limit, price target hit, anomaly detected), they generate events that LLM agents process. The agent's role is strategic reasoning and recommendation, not execution speed. This complements latency-sensitive pipelines: keep the microsecond path deterministic, hand off to the agent only once an alert is raised.

Match agent capabilities to task requirements. Speed-critical tasks need traditional algorithms; reasoning-critical tasks need agents.

Integration Patterns: External feeds reach agents through **webhooks** (HTTP callbacks for immediate notification) or **message queues** (durable event streams with delivery guarantees). 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. 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.

2.2 Human Prompts as Events

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

Chat interfaces are the most direct channel. The associate types "Find Fifth Circuit authority on personal jurisdiction for e-commerce defendants," the agent searches and presents summaries, and the associate follows up with refinements. The analyst asks for revenue growth comparisons across portfolio companies, receives a table, and requests additional filtering. Chat enables iterative clarification while maintaining architectural consistency: each message is simply an event processed through the standard agent loop, with tighter latency expectations than background tasks.

Email routing enables agents to process work arriving through existing communication channels. A general counsel forwards a business unit's compliance question to an agent mailbox; the agent extracts the question, searches relevant guidance, and emails back an assessment. The challenge is intent classification: email bodies are unstructured and may include forwarded threads with multiple topics.

Collaboration platforms like Slack and Teams allow agents to appear as team members. Users @mention the agent in channels, send direct messages, or use slash commands. The litigation team discussing strategy can invoke research directly in their coordination channel. Security requires authorization checks at the agent layer, since collaboration platforms may log responses and channels may include unauthorized viewers.

Voice interfaces work best for short, urgent requests where typing is impractical. They introduce transcription errors (legal jargon like "Chevron deference" may transcribe incorrectly) and authentication challenges. High-stakes voice requests should require explicit confirmation before execution.

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, annual filings. For these recurring tasks, time itself triggers execution.

Calendar-driven deadlines govern legal practice. Answer the complaint within 21 days. File motions 30 days before hearings. Respond to discovery within 30 days. Agents can monitor litigation calendars, calculate deadlines accounting for court holidays, schedule reminders as deadlines approach, and escalate if work remains incomplete. Sophisticated deadline agents go further, retrieving the complaint, extracting claims, generating draft answers with standard defenses, and presenting drafts for attorney review before filing. Financial institutions face similar deadline-driven work, from SEC reporting deadlines to tax filings to contractual obligations to lenders.

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

End-of-day workflows in financial institutions reconcile trades, calculate valuations at market close, generate P&L reports, and prepare risk reports for the next morning. At market close, an EOD agent retrieves final prices, marks positions to market, calculates P&L, and identifies unexplained variances. The agent then distributes reports to stakeholders. If any step fails, the agent escalates rather than proceeding with incomplete data. Law firms run similar periodic workflows, reminding attorneys to enter time, generating draft invoices at month-end, and flagging anomalies for partner review.

2.4 Escalation Events: When Agents Reach Their Limits

The previous three channel types bring work into the agent system from outside. Escalation events operate internally: the agent generates an event signaling it has reached a limit and requires human intervention, transferring control to human decision-makers when the agent cannot proceed autonomously.

Four escalation triggers appear most frequently:

Budget exhaustion: The agent approaches resource limits (token consumption, iteration counts, time limits, or cost caps) and must decide whether to stop or request additional budget.

Low confidence: Uncertainty is too high for autonomous action. Conflicting authority, novel situations, or results that seem implausible warrant human review.

Approval requirements: Certain actions require explicit human authorization regardless of the agent's confidence: filing court documents, sending client communications, executing large trades.

Errors and anomalies: Tools fail repeatedly, data is inconsistent, or the agent detects red flags that require human investigation.

Section 9 provides comprehensive treatment of when and how agents should escalate to humans.

2.5 Event Routing and Prioritization

With events arriving from multiple channels, agents need routing and prioritization logic. A law firm routes work similarly. Client calls go to appropriate attorneys, court filings route to the litigation coordinator, research requests go to assigned associates. Agent systems implement the same pattern through a central router that receives events, examines metadata, applies routing rules, and dispatches to appropriate handlers.

Routing rules map event attributes to handlers. Court filing notifications for Matter 12345 route to that matter's litigation agent. SEC filings by portfolio companies route to the monitoring agent. Routing can be static (predefined rules) or dynamic (classifiers that analyze content and identify topics). For multi-agent architectures, routing determines delegation: an orchestrator receives high-level tasks, classifies them, and routes to specialist agents.

Priority queues implement tiered processing. Urgent events (emergency motions, margin calls) enter the high-priority queue and are processed immediately, potentially interrupting lower-priority work. Routine tasks enter standard queues. Background work (database updates, model retraining) runs when resources are idle. Priority can be rule-based (certain event types always urgent) or adaptive (priority escalates as deadlines approach).

Temporal constraints require processing within specific windows. Court filings have deadlines, trading must occur during market hours, EOD reports must complete before the next morning. Agents track these constraints, calculate time remaining, and escalate priority as deadlines approach.

Overload management prevents cascading failures when events arrive faster than processing capacity. Rate limiting caps how many events agents accept per minute, protecting downstream APIs. Backpressure signals upstream systems to slow down. Load shedding drops low-priority work to preserve capacity for critical tasks during peak demand. During a market crash, trade execution and risk calculations take precedence; routine reporting can wait.

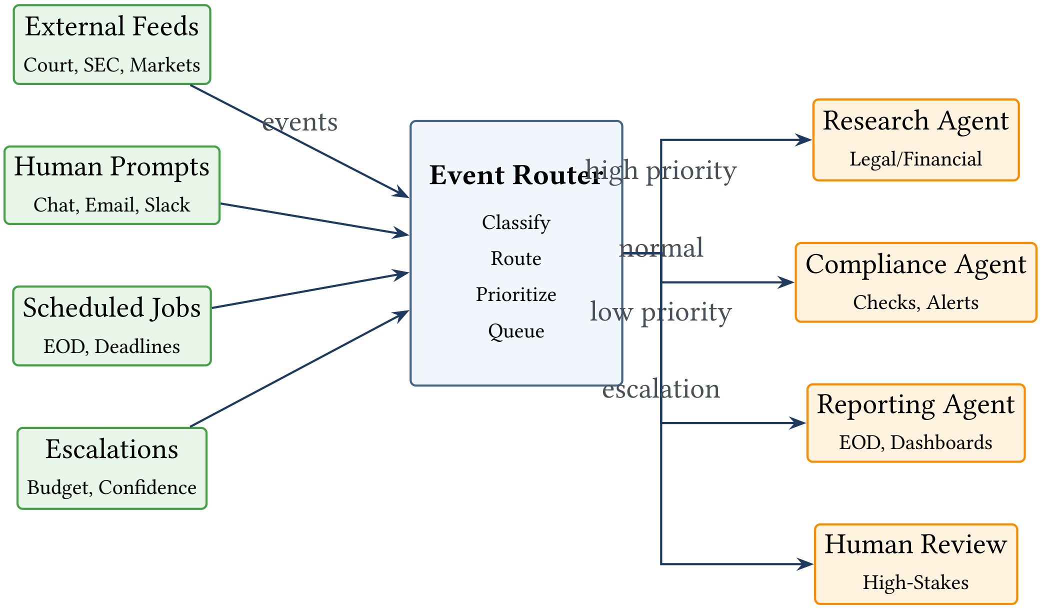


Figure 2: Event routing architecture showing how events from multiple channels flow through a central router that classifies, prioritizes, and dispatches to appropriate handlers.

2.6 Surfaces: How Users Experience Agent Systems

The same underlying architecture can manifest through different user interfaces, or *surfaces*. Understanding surfaces matters because the appropriate surface depends on the task, the user's expertise, and how the output will be used. Three primary surfaces serve different purposes. Chat surfaces suit interactive exploration, where the partner thinking through case strategy or the analyst exploring market conditions remains actively engaged, refining direction through dialogue. Automation surfaces suit continuous monitoring such as portfolio surveillance, docket tracking, and compliance alerts, where the agent works in the background and users receive outputs only when relevant. Document surfaces suit defined deliverables like research memos, due diligence reports, and client presentations, where the agent produces work products for human review and editing before distribution.

Most deployments combine these surfaces in practice: chat for ad hoc queries and exploratory thinking, automation for continuous monitoring and alerting, and document generation for formal deliverables that must be filed, sent to clients, or presented to committees. The underlying agent architecture supports all three; the surface simply determines how users encounter the system in their day-to-day work.

2.7 Evaluating Trigger Systems

When evaluating agent systems, whether building or buying, assess trigger capabilities against five criteria:

Coverage: Does the system receive events from all relevant sources? A litigation agent that monitors

CM/ECF but not state court dockets has incomplete coverage.

Latency: How quickly do events reach the agent? Real-time market data requires sub-second delivery; docket alerts can tolerate minutes.

Reliability: What happens when feeds fail? Systems need retry logic, fallback sources, and alerting when data goes stale.

Priority mechanisms: Can the system distinguish urgent from routine? During a market crash or litigation crisis, the right events must reach the right handlers immediately.

Auditability: Is every trigger logged? When a regulator asks why the agent took action, you need a complete record of the triggering event.

2.8 From Triggers to Action

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

- **Understand intent** (Q2, Section 3): What is being asked?
- **Perceive information** (Q3, Section 4): What does the agent need to know?
- **Take action** (Q4, Section 5): What should the agent do?
- **Remember context** (Q5, Section 6): What should persist across sessions?
- **Plan execution** (Q6, Section 7): How should work be decomposed?
- **Recognize completion** (Q7, Section 8): When is the task done?
- **Escalate when needed** (Q8, Section 9): 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 agent. The agent 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 agent 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 agent 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.

Agent 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?*

Instruction, Intent, Goal, and Task

Instruction: The words the user provides (e.g., "Review this credit agreement").

Intent: The underlying purpose, constraints, and success criteria behind the instruction (e.g., "identify material risks to the lender by tomorrow").

Goal: The desired end state the intent points to (e.g., "produce a lender-risk memo that meets policy"), as defined in the GPA framework.

Task: The concrete unit of work the agent will execute to advance the goal (e.g., "extract covenants and compare to template").

Intent bridges instruction to goal and shapes the tasks the agent will plan.

3.1 From Instruction to Intent

Consider three real instructions a legal or financial professional might give:

- "Review this credit agreement for risks"
- "Rebalance to reduce tech exposure"
- "Look into the Johnson matter"

Each is ambiguous. The word "risks" raises immediate questions about perspective (lender or borrower), scope (material risks or all risks), and domain (legal, financial, or both). "Reduce tech exposure" leaves open the target level, the mechanism (sales, hedges, or both), and the tax and timing constraints. "Look into" specifies neither depth, urgency, nor deliverable format. The instruction is clear enough to start; the intent is not.

Pre-LLM systems handled intent through rigid parsing: keyword matching, slot filling, decision trees. 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 intents but look nothing alike to a keyword matcher. The gap between instruction and intent has always existed; what has changed is our ability to bridge it.

Large language models dramatically improved the ability to infer intent 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 intent, 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 communicate through implication—"This needs to be right" signals high stakes; "When you get a chance" signals low urgency—and these signals may not be explicitly parsed. Research has shown that LLMs can "exploit loopholes" by selectively misunderstanding ambiguous requests in ways that appear helpful but avoid difficult work. Governance must monitor for this failure mode.

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.

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

- **Information retrieval**: "What's the current NAV?" "Find the latest 10-K"
- **Research and analysis**: "Research whether we can pierce the corporate veil"
- **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 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.

Entity and constraint recognition: Beyond classification, the agent must extract entities (what the task concerns) and constraints (what bounds apply):

Entities: Matters, clients, securities, parties, documents, jurisdictions. "Review the Smith acquisition agreement" references a specific document; "Research Delaware fiduciary duties" references a jurisdiction.

Temporal constraints: Deadlines, as-of dates, time windows. "By Friday" sets a deadline; "as of year-end 2024" sets a reference date; "over the past quarter" defines a window.

Resource constraints: Budget limits, scope bounds. "Spend no more than 2 hours" limits effort; "focus on Articles 3 and 4" limits scope.

Format constraints: Deliverable specifications. "Summarize in one page" constrains length; "prepare a memo for the file" specifies format; "I need something to show the client" signals external audience.

Audience and privilege constraints: Who will see the output (internal team, client, regulator) and what privilege or confidentiality must be preserved.

Risk and compliance constraints: Limits that may not be stated but must be inferred. A compliance review implicitly requires flagging violations; a client communication implicitly requires privilege protection.

Structured goal representation: Extracted components can be organized into structured representations that guide execution. This structured representation resembles a short assignment memo that the planning system can act on (Section 7) and the termination system can measure against (Section 8):

```
task_type: document_review
document: Smith Acquisition Agreement
objective: identify change-of-control provisions
constraints:
  deadline: 2025-01-15
  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.

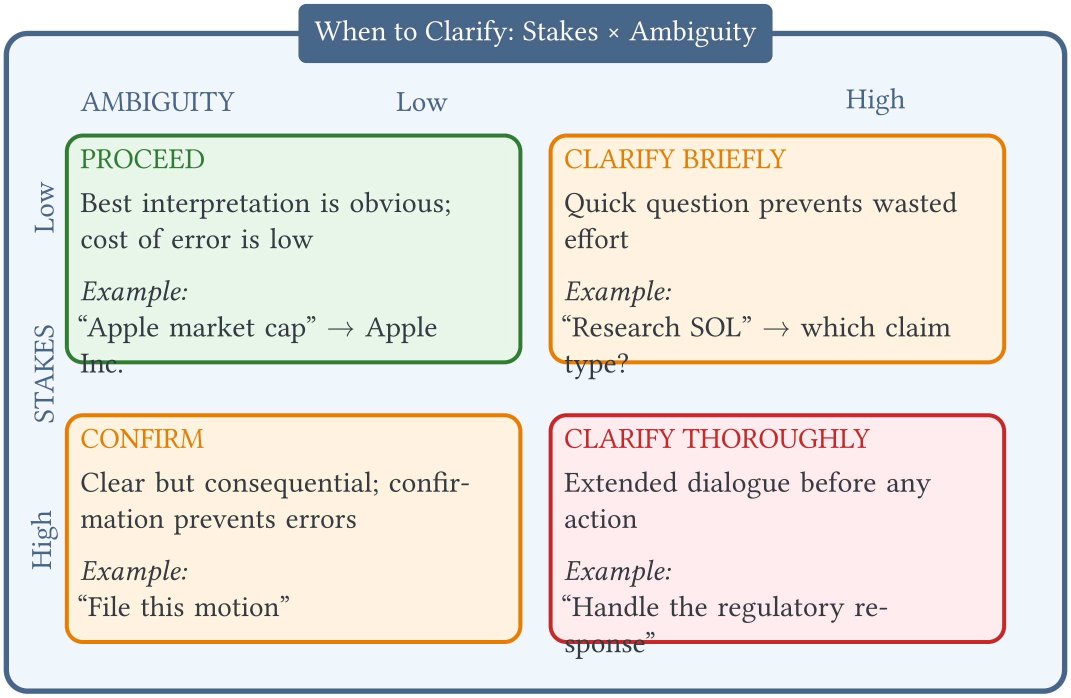
When to clarify: The decision to clarify depends on ambiguity severity and action stakes:

Low stakes, low ambiguity: Proceed with best interpretation. If the user asks “What’s Apple’s market cap?” and you’re unsure whether they mean Apple Inc. or Apple Hospitality REIT, the dominant interpretation is obvious and the cost of being wrong is low (easy to correct).

Low stakes, high ambiguity: Clarify briefly. If the user asks “Research the statute of limitations” without specifying the claim type, a quick clarification prevents wasted effort.

High stakes, low ambiguity: Confirm before acting. If the instruction is clear but consequential (“File this motion”), confirmation prevents irreversible errors.

High stakes, high ambiguity: Clarify thoroughly. If the user says “Handle the regulatory response” for a complex matter, extended clarification is appropriate before taking any action.



When to Clarify: Stakes × Ambiguity

STAKES / AMBIGUITY	Low	High
Low	PROCEED Best interpretation is obvious; cost of error is low *Example:* “Apple market cap” → Apple Inc.	CLARIFY BRIEFLY Quick question prevents wasted effort *Example:* “Research SOL” → which claim type?
High	CONFIRM Clear but consequential; confirmation prevents errors *Example:* “File this motion”	CLARIFY THOROUGHLY Extended dialogue before any action *Example:* “Handle the regulatory response”

Figure 3: Decision framework for when agents should clarify user intent. Stakes measure consequence of error; ambiguity measures interpretation confidence. High-stakes or high-ambiguity tasks warrant clarification despite potential for slowing response.

How to clarify: Effective clarification is specific, contextual, and actionable:

Specific: “Which jurisdiction’s statute of limitations—Delaware or New York?” not “Can you clarify?”

Contextual: Reference 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?”

Actionable: Offer 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?”

Bounded: Limit 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?”

The Default Interpretation Risk

Research has documented that LLMs sometimes select a default interpretation rather than asking for clarification, even when ambiguity is significant. This “proceed without asking” behavior can be problematic:
The risk: The agent interprets “review the contract” as a surface-level summary when the user expected deep issue-spotting. Work product is delivered, but it is wrong.
Mitigation strategies:

- Prompt engineering that emphasizes clarification for ambiguous requests
- Confidence thresholds that trigger clarification below a certainty level
- User training to provide detailed initial instructions
- Checkpoint reviews before significant work begins

Governance implication: Monitor for cases where the agent proceeded confidently but delivered unexpected results. These may indicate calibration problems in ambiguity detection.

3.4 Constraint Identification

Beyond explicit instructions, agents must identify constraints that bound acceptable execution.
Temporal constraints: Deadlines and time windows. Some are explicit (“by Friday”); others are implicit (court filing deadlines calculated from rules); still others are contextual (“before the board meeting” requires knowing when the meeting is).

Resource constraints: Budget and effort limits. Token budgets limit API costs; time budgets limit calendar impact; scope constraints focus effort on high-value areas.

Scope constraints: 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: How deliverables should appear. Memo versus email versus presentation; formal versus casual tone; internal versus client-facing.

Risk and compliance constraints: What must be avoided. Privilege protection, conflicts of interest, regulatory restrictions, confidentiality obligations. These constraints often apply implicitly based on context.

Inferring implicit constraints: Professionals operate under 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 validation:

Reflection and summarization: The agent restates its understanding before proceeding, giving the user an opportunity to correct misunderstandings before work begins.

Chunked validation: For complex tasks, validate in phases rather than all at once. After completing research, summarize findings and confirm direction before drafting. After drafting, confirm the approach before finalizing. Each checkpoint prevents error propagation.

Confidence signaling: The agent should 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.

Legal example—credit agreement review: 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?" The extracted goal:

```
{
  "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."

Financial example—portfolio rebalancing: 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 and long-term gains over short-term). The extracted goal:

```
{
  "task_type": "portfolio_rebalancing",
  "objective": "reduce_tech_exposure",
  "target": "25%",
  "mechanism": "sales",
  "constraints": {
    "tax_optimization": true,
    "prefer_loss_harvesting": true
  },
  "approval_required": true
}
```

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 (Section 11) 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 (Q1) and intent extraction revealing what's being asked (Q2), the agent needs capabilities to gather information and effect change. Section 4 examines the next question: how does an agent find things out?

4 How Does an Agent Find Things Out?

A junior associate's effectiveness depends not just on reasoning ability but on access. The ability to query Westlaw, access Bloomberg terminals, and search the firm's precedent database determines what problems the associate can actually solve. Without access to information sources, even the most capable professional reasons in a vacuum.

Agent systems 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 library card, the database subscription, and the research assistant combined: the mechanisms 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 framework; it answers the question 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. The distinction matters for governance, because read operations carry different risks than write operations.

4.1 Perception Tool Categories

Different tasks require different tools for gathering information, and effective perception depends on having the right tool for the purpose at hand.

Information Retrieval Tools: For gathering authoritative information, professionals use research platforms. **Legal research tools** include Westlaw, Lexis, Bloomberg Law, PACER for federal court filings, state court docket systems, and regulatory databases like EDGAR. An agent with these tools can search case law, retrieve opinions, check citator status, and download filings.

Financial research tools include Bloomberg Terminal, Reuters Eikon, FactSet, and proprietary analytics platforms. An agent with these tools can query real-time prices, retrieve fundamentals, access analyst research, and pull historical data.

Internal knowledge bases include the firm's document management system (iManage, NetDocuments), precedent databases, deal archives, and research memo repositories. An agent with access can retrieve prior work product, find template language, and check how the firm handled similar matters.

Document Processing Tools: Raw documents need processing before they are useful:

Text extraction converts PDFs, scanned documents, and images into searchable text. OCR tools handle scanned filings; PDF parsers extract text from native documents; table extractors preserve structure from financial statements.

Document classification identifies document types. In due diligence, an agent processing a data room needs to distinguish contracts from correspondence, financial statements from presentations. Classification enables appropriate routing.

Entity extraction identifies parties, dates, amounts, and other structured data from unstructured documents. Extracting the borrower name, facility amount, and maturity date from a credit agreement enables structured analysis.

Computation Tools: Some forms of perception require calculation rather than simple lookup:

Deadline calculators determine response dates from rules. Federal Rules require answers within 21 days—but calculating from service date, accounting for holidays, and applying local rules requires computation.

Citation formatters convert case information into proper Bluebook format. Financial equivalents normalize identifiers (CUSIP, ISIN, ticker) across different systems.

Risk metrics calculate exposure, VaR, duration, or other quantitative measures from position data. These computations inform reasoning without changing portfolios.

4.2 Model Context Protocol (MCP)

The Model Context Protocol standardizes how agents access tools. Before standardization, every database had different commands and output formats: Westlaw worked one way, Lexis another, Bloomberg a third. MCP creates a common interface: learn the protocol once, access any compatible tool.

MCP Architecture: The architecture has three roles:

MCP Host: Manages the agent and controls which tools it can access. Like the firm's IT system determining database subscriptions.

MCP Client: The agent-side component that discovers and uses tools.

MCP Server: A tool exposing capabilities through a standardized interface. The document management system, internal knowledge base, or custom legal research tool each runs as an MCP server.

Communication follows a simple pattern: servers publish manifests declaring capabilities; clients connect through hosts; clients send structured requests; servers return structured results.

MCP Resources: For perception, MCP defines **Resources**—read-only data access endpoints. Resources let agents:

- Query case law databases and receive structured results
- Retrieve documents from management systems
- Access market data feeds
- Search internal knowledge bases
- 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 M×N Problem

Without MCP: 10 agents × 10 tools = 100 custom integrations.
With MCP: 10 agents + 10 tools = 20 implementations (each learns the protocol once).
Legal: One agent queries document management systems, internal knowledge bases, and custom research tools through the same protocol.
Financial: One agent accesses portfolio systems, risk engines, and compliance databases through the same protocol.
As of late 2025, over 7,260 MCP servers have been catalogued in community directories. The ecosystem provides ready-made integrations for common tools.

4.3 Memory as Perception into Institutional Knowledge

Memory systems (Section 6) serve as perception tools into institutional knowledge. When an agent queries the firm's precedent database, it perceives accumulated expertise:

Retrieval-Augmented Generation (RAG) enables semantic search over document archives. The agent doesn't just keyword-match; it finds conceptually similar content. A search for "breach of fiduciary duty" retrieves documents about "violation of trust obligations" even if exact words differ.

Vector stores power this semantic search by encoding documents as high-dimensional embeddings. The technology enables perception into large knowledge bases that would be impractical to load into context.

Prior work product becomes accessible through memory. When starting a new registration statement, the agent can perceive prior S-1 filings, SEC comment histories, and successful disclosure language. This institutional knowledge informs current work.

Memory-as-perception distinguishes experienced agents from novices. The junior associate reasons from first principles; the senior associate draws on pattern recognition from hundreds of matters. Memory gives agents access to accumulated experience.

4.4 Domain-Specific Perception Requirements

Perception for regulated professional services requires specialized enhancements:

Authority and Provenance. Not all information is equally authoritative. Perception systems must track provenance:

Authority weighting: Primary authority (statutes, binding precedent) should rank higher than secondary sources. When searching for "insider trading liability," a Supreme Court opinion should outrank a law review note using more similar language.

Source verification: Did this case actually come from Westlaw, or was it hallucinated? Perception tools must return verifiable sources that can be independently checked.

Currency validation: Is this authority still good law? Citator integration validates that retrieved cases haven't been overruled.

Jurisdiction and Scope. Legal and regulatory information is bounded by jurisdiction:

Jurisdiction awareness: California precedent doesn't bind Texas courts; SEC rules differ from CFTC rules. Perception must respect jurisdictional boundaries and filter appropriately.

Temporal validity: Law changes. Perception systems must track effective dates. Financial temporal validity varies by context: milliseconds for trading prices, quarters for compliance effective dates.

Identifier resolution: Citations appear in multiple formats ("123 F.3d 456" and "123 F3d 456" are the same case). Financial identifiers proliferate, including ticker symbols, CUSIP numbers, ISIN codes, and Legal Entity Identifiers. Perception must normalize identifiers to enable consistent retrieval.

Matter and Client Isolation. Most critically, perception must respect confidentiality boundaries:

Matter isolation: An agent working on Matter A cannot perceive documents from adverse Matter B. Ethical walls must be enforced at the perception layer.

Client isolation: In financial contexts, an agent advising Client X cannot perceive material non-public information from Client Y's engagement.

Audit trails: Every perception must be logged—what was accessed, by which agent, for which matter. This enables compliance review and breach detection.

See Section 6 for detailed treatment of isolation requirements in memory systems.

4.5 Tool Design Principles

Good perception tools follow design principles that enable reliable operation:

Single Responsibility. Each tool should do one thing well. A poorly designed tool bundles multiple functions:

`legal_research(query, format, validate, extract)` searches Westlaw, formats citations, validates authority, and extracts holdings: four functions in one interface. When it fails, you can't tell which step failed.

A better approach separates tools by function. `search_cases(query, jurisdiction)` returns citations. `retrieve_case(citation)` fetches text. `shepardize(citation)` checks validity. `format_citation(data, style)` converts to Bluebook. The agent composes them; failures are isolated.

Graceful Failure. When things go wrong—and in production, things always go wrong—tools should return informative errors:

Poor: `Exception: NullPointerException at line 847`

Good: `Error: Case not found for citation “123 F.3d 456”. Case may not be in database. Suggestion: Check citation manually or try alternative reporter.`

The first tells you nothing. The second explains what happened and suggests recovery. In legal work, graceful failure is how you avoid malpractice: when you cannot find authority, report that explicitly rather than proceeding silently.

Least Privilege: Perception tools should request only the permissions they need. A legal research tool needs read access to case databases, not write access to the document management system. If a compromised agent gains perception credentials, damage is limited to what those credentials allow.

Rate Limiting: Agents can get stuck in perception loops—searching repeatedly without progress. Tools should track invocation frequency and refuse requests beyond reasonable thresholds. If the agent has searched five times with no results, it should stop and escalate rather than continuing indefinitely.

4.6 Evaluating Perception Capabilities

When evaluating agent systems, assess perception against these criteria:

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

Retrieval quality: Verify that the agent finds relevant information by testing with known-good queries where you know what should be retrieved. Measure both precision (relevance of results) and recall (completeness of relevant results).

Authority and provenance: Confirm that the system distinguishes authoritative from secondary sources, that you can trace retrieved information to its source, and that citations are independently verifiable.

Access controls: Verify that permissions are appropriate, that the agent can access only what it should, and that confidentiality boundaries are enforced across matter and client lines.

Failure handling: Assess the agent’s behavior when perception fails: whether it retries, tries alternatives, or escalates appropriately rather than crashing or proceeding with incomplete information.

Audit capability: Confirm that every perception is logged and that you can reconstruct what information the agent accessed during a task for compliance review and post-hoc analysis.

4.7 From Perception to Action

Perception enables agents to gather information, but agents must also be able to effect change—file documents, send communications, execute trades. The critical distinction is simple but important:

Perception tools are read-only. They observe without changing the world. If a perception tool fails or returns wrong results, no external state has changed; you can retry or try alternatives.

Action tools change state. They file documents, send emails, and execute trades. Once executed, some actions cannot be undone. The risks are different, and the governance must be different as well.

Section 5 examines the next question: how does an agent make things happen?

5 How Does an Agent Make Things Happen?

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

Agent 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, execute transactions. Action implements the "A" in the GPA 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, update databases. Once executed, some actions cannot be undone.
The distinction between perception and action is fundamental to governance. Perception risks include accessing wrong information or missing relevant data. Action risks include taking wrong actions that harm clients, violate regulations, or create liability.

5.1 Action Tool Categories

Action tools vary in consequence. The key dimension is **reversibility**: can the action be undone if something goes wrong?

Communication Tools (Partially Reversible): Communication tools send information to others:

Internal communications: Emails to colleagues, messages in collaboration platforms, updates to internal systems. These are partially reversible: you can follow up with corrections, but you cannot unsend.

External communications: Emails to clients, letters to opposing counsel, regulatory notifications. Higher stakes than internal; recipients outside your control. Retractions are possible but create their own problems.

Automated alerts: System-generated notifications, compliance alerts, deadline reminders. Often templated with limited customization.

Governance implication: Internal communications may proceed with post-hoc review; external communications typically require pre-approval.

Document Management Tools (Largely Reversible): Document management tools create and organize work product:

Document creation: Drafting memos, generating reports, producing analysis. The documents exist internally and can be revised before distribution.

Document organization: Filing documents in management systems, tagging and categorizing, maintaining matter files. Generally reversible through re-organization.

Template application: Generating documents from templates, populating forms, producing standard documents. Low-risk if templates are validated.

Governance implication: Document creation is relatively low-risk; documents can be revised before external sharing. Validation before distribution is the key control.

Filing and Submission Tools (Largely Irreversible): Filing tools submit documents to external authorities:

Court filings: E-filing through CM/ECF or state systems. Once filed, documents become part of the public record. Amendments are possible but do not erase the original.

Regulatory submissions: SEC filings through EDGAR, FINRA submissions, state regulatory filings. Subject to regulatory requirements; errors can trigger enforcement.

Contract execution: Signature, execution, delivery of binding agreements. Creates legal obligations that may be difficult or impossible to unwind.

Governance implication: Filing and submission require mandatory pre-approval. The irreversibility demands human verification before execution.

Transaction Execution Tools (Irreversible or Costly): Transaction tools execute binding transactions:

Trade execution: Entering orders, executing trades, confirming allocations. Once executed, trades settle and create positions. Reversal requires offsetting trades at market prices.

Payment processing: Wire transfers, payment initiation, fund disbursements. Once sent, funds are gone. Recovery requires recipient cooperation.

System updates: Modifying production databases, updating client records, changing configurations. May affect downstream systems; reversal may be complex.

Governance implication: Transaction execution requires the strictest controls—multi-factor approval, segregation of duties, real-time monitoring.

5.2 The Reversibility Framework

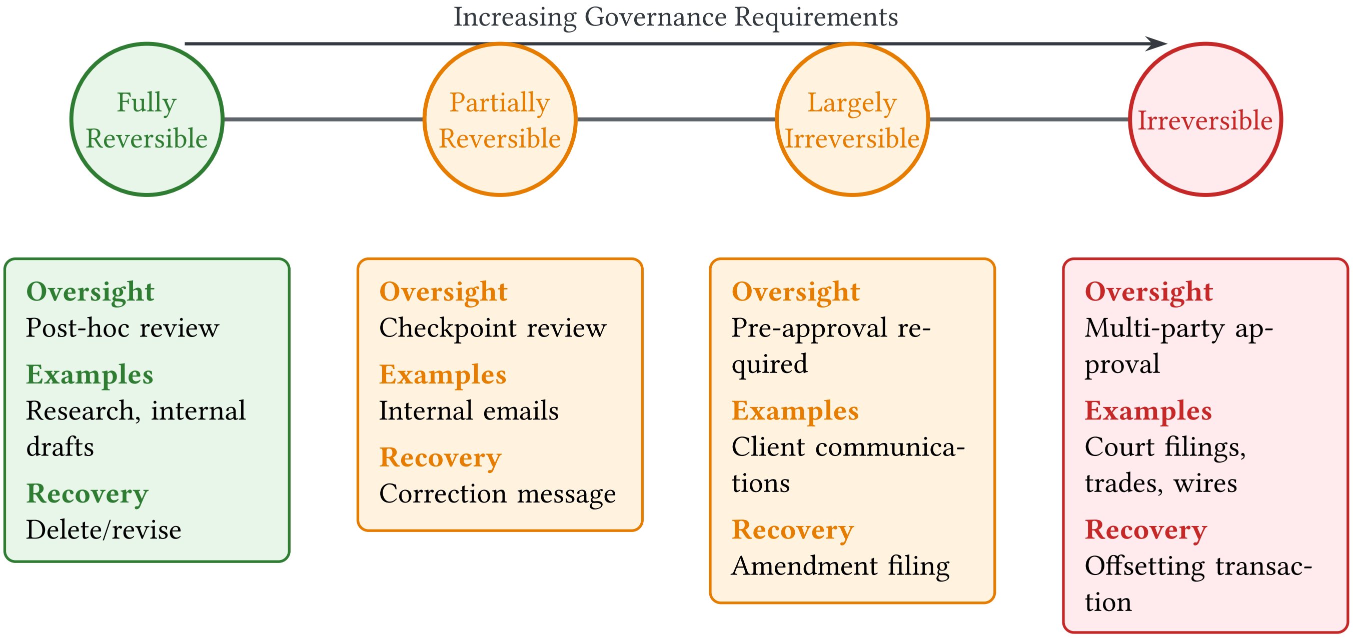


Figure 4: Action reversibility spectrum and corresponding governance requirements. As actions become less reversible, oversight shifts from post-hoc review to pre-approval and multi-party authorization. Recovery mechanisms range from simple deletion for fully reversible actions to complex offsetting transactions for irreversible ones.

Reversibility determines required oversight. Consider how you delegate to a junior associate:

Fully reversible actions (research, internal drafts): The associate works independently. If they make mistakes, you catch them in review. No external harm occurs.

Partially reversible actions (internal communications, document organization): Checkpoint review. The associate completes work; you review before it affects others significantly.

Largely irreversible actions (client communications, filings): Pre-approval required. The associate prepares; you approve before execution.

Irreversible actions (trade execution, fund transfers): Multi-party approval with controls. Multiple people must verify before execution.

Agent governance should track reversibility. The architecture should enforce appropriate controls based on action classification.

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)

5.3 MCP Tools and Prompts for Action

The Model Context Protocol defines two capability types relevant to action:

MCP Tools: MCP Tools are executable functions that may change state. Unlike Resources (read-only), Tools can:

- Create and modify documents
- Send communications
- Submit filings
- Execute transactions
- Update external systems

Tool manifests should include risk metadata: reversibility classification, approval requirements, audit requirements. This enables hosts to enforce appropriate controls.

MCP Prompts: MCP Prompts are reusable templates for common tasks. For action tools, prompts encode standard operating procedures:

Legal examples: Contract review checklist prompts, due diligence workflow prompts, filing preparation prompts.

Financial examples: Trade compliance check prompts, client onboarding prompts, regulatory submission prompts.

Prompts standardize action sequences, reducing variation and error. They are particularly valuable for high-stakes actions where consistency matters.

5.4 Action Security

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

Core Controls: All action tools must implement:

Authentication: Verify the agent is who it claims to be. Service accounts with strong credentials; no shared or default passwords.

Authorization: Verify the agent has permission for this specific action. Role-based access control; principle of least privilege.

Input validation: Reject malformed or suspicious requests. Validate all parameters before execution.

Output confirmation: For high-stakes actions, require confirmation before execution completes.

Rate limiting: Cap action frequency to prevent runaway execution. Escalate after repeated actions.

Audit logging: Record every action with full context: agent identifier, action name, parameters, timestamp, result, matter/client context.

Threat-Specific Mitigations: **Prompt injection through action parameters**: Adversaries embed instructions in parameters the agent passes to action tools. *Mitigation*: Sanitize all parameters; never pass raw user input directly to action interfaces; validate parameter formats against strict schemas.

Privilege escalation through tool chaining: An agent chains multiple tools to achieve capabilities no single tool grants. *Mitigation*: Analyze tool combinations for escalation paths; require human approval for tool sequences that span security boundaries.

Action replay: A captured action request is replayed to re-execute the action. *Mitigation*: Implement nonces or timestamps; reject duplicate requests; maintain action logs for detection.

5.5 Approval Workflows

For non-reversible actions, the agent prepares; humans approve. Several patterns implement this:

Single Approver: The agent completes preparation and presents to one designated approver. Appropriate for routine actions with clear approval authority.

Pattern: Agent prepares court filing → presents to supervising attorney → attorney reviews and approves → agent submits.

Multi-Party Approval: High-stakes actions require multiple independent approvers. Appropriate for actions with significant financial or legal exposure.

Pattern: Agent prepares wire transfer → operations reviews amounts and accounts → compliance reviews purpose and restrictions → manager provides final approval → agent executes.

Escalating Approval: Approval authority escalates with transaction size or risk. Routine actions have lower approval thresholds; exceptional actions escalate to senior personnel.

Pattern: Trades under $100K → desk manager approval. Trades $100K-$1M → senior trader approval. Trades over $1M → CIO approval.

Approval Request Design: Effective approval requests include:

- Clear description of the proposed action
- Context: why is this action needed?
- Risk assessment: what could go wrong?
- Reversibility: can this be undone?
- Supporting evidence: what analysis supports this action?
- Deadline: when is approval needed?

The approver should be able to make an informed decision from the request alone, without needing to investigate further.

5.6 Rate Limiting and Circuit Breakers

Agents can get stuck in action loops: submitting the same request repeatedly, sending multiple messages, attempting failed transactions again and again. Controls prevent runaway execution:

Rate Limiting: Cap how many actions the agent can take per time period:

Per-action limits: No more than 5 emails per minute; no more than 10 trades per hour.

Per-matter limits: No more than 20 actions on any single matter per day without human review.

Cost limits: No more than $1,000 in transaction costs per session.

When limits are reached, the agent pauses and escalates rather than continuing.

Circuit Breakers: Automatic stops when anomalies are detected:

Repeated failures: If the same action fails 3 times, stop and escalate. Do not retry indefinitely.

Unusual patterns: If action rate suddenly spikes, pause for review. May indicate agent malfunction or compromise.

Threshold breaches: If cumulative actions exceed daily limits, stop automatically. Resume requires human authorization.

Circuit breakers transform potential runaway failures into controlled pauses that allow human intervention.

5.7 Evaluating Action Capabilities

When evaluating agent systems, assess action capabilities against these criteria:

Action inventory: What actions can the agent take? Map available action tools against workflow requirements.

Reversibility classification: Is each action properly classified? Are controls appropriate to reversibility level?

Approval workflows: Are approval gates implemented for non-reversible actions? Do approvers receive sufficient information?

Security controls: Are authentication, authorization, and audit logging implemented? Have penetration tests been conducted?

Rate limiting: Are limits in place? Do they match acceptable risk tolerances?

Rollback capability: What happens when actions fail? Are recovery procedures documented and tested?

5.8 From Action to Governance

Action tools are where agent systems create real-world consequences. The governance implications are significant:

Perception risks (Q3) include accessing wrong or incomplete information. The agent reasons from bad data, but no external harm has occurred yet.

Action risks (Q4) include taking wrong actions that harm clients, violate regulations, or create liability. The consequences are external and may be irreversible.

The following chapter examines action governance in detail: actuation controls that limit what agents can do, approval gates that require human verification, and audit trails that enable accountability.

Within this chapter, Section 9 examines when agents should *not* act—recognizing situations that require human decision-making rather than autonomous execution. The interplay between action capability and escalation judgment is central to safe agent deployment.

Section 6 examines the next question: how does an agent remember things? Memory enables agents to maintain context across sessions, learn from experience, and access institutional knowledge.

6 How Does an Agent Remember Things?

Every experienced legal professional knows that institutional memory makes the difference between efficient work and reinventing the wheel. When you start a new securities registration matter, you do not begin from scratch. You pull the last three S-1 filings the firm completed, review the SEC comment history, and check the precedent database for disclosure language addressing similar risk factors. You do not re-research basic questions like "What are the disclosure requirements for executive compensation?" The firm maintains templates and form language 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 from scratch. You pull the research file, review your prior DCF model and industry analysis, then update assumptions with recent earnings data and sector trends.

The accumulated research—organized and accessible—enables incremental refinement rather than duplicative work.

Memory in agent systems serves the same purpose: context retention across sessions and learning from experience. Without memory, every interaction starts fresh. The agent does not remember what it researched yesterday, what approaches worked, or what the human told it about case strategy. With memory, the agent maintains continuity, much like the case file that follows a matter from initial consultation through trial.

Agent Memory

Agent memory stores and retrieves information across timescales. Short-term memory is the documents spread across an associate's desk during active work. Long-term memory is the firm's knowledge management system with decades of research memos. Episodic memory is the case file that tracks what happened on this specific matter. Semantic memory is the legal principles every attorney internalizes over their career.

6.1 Memory Types: From Desk to Archive

Law firms use layered filing systems, each suited to different timescales and access patterns. The associate's desk holds today's active work, the matter file contains everything related to this engagement, and the firm's precedent database archives decades of institutional knowledge. Each layer trades immediacy for capacity, and effective practice requires knowing which system to consult when.

Working Memory (Context Window): The associate has papers spread across their desk: the documents actively in use right now. This immediate context is **working memory**. In agent systems, working memory takes the form of the *context window*, the tokens currently loaded in the LLM's attention.

Just like desk space, context windows have strict limits. The associate can only have so many documents open at once; the agent can only hold so many tokens in active context (as of late 2025, 200K tokens for leading models, though this ceiling continues to rise). When the case involves more documents than fit on the desk, you need other storage systems.

The banker has *market data on the trading screen*: live prices, recent news, positions from today's session. This too is working memory: fresh and immediately accessible but gone when the session ends.

Episodic Memory: Beyond the desk sits the matter file for the specific engagement. Every memo, every piece of correspondence, every research result related to this matter goes in the file. The associate does not re-research questions already answered; the file comes first. When the partner asks "What's our argument on venue?," the associate pulls the file and reads the prior research memo rather than starting over.

Agent systems call this **episodic memory**: the history of actions and outcomes for a specific task or session. The agent remembers: "I searched for Ninth Circuit venue cases, found three relevant opinions, drafted analysis, partner reviewed and approved." When asked a follow-up question, the agent retrieves that prior work.

The financial parallel is the research file for each position. When you revisit a stock you analyzed six months ago, you do not rebuild the entire investment thesis. You pull the file, read your prior analysis, and update it with new information. The agent does the same: retrieve prior analysis, check what has changed, update conclusions.

Retrieval-Augmented Generation (RAG): The third layer is the firm's precedent database: institutional knowledge accumulated over decades. Every time the firm handles a particular type of matter, the work product goes into the archive. When you need language for a force majeure clause in a construction contract, the precedent database offers fifty examples from prior deals. When you need briefing on qualified immunity, the database contains the firm's best arguments from the past ten years, organized by circuit and issue.

Agent systems implement this through **retrieval-augmented generation (RAG)**: dynamically fetching relevant information from a large corpus to augment the agent's reasoning. RAG enables agents to access institutional knowledge beyond what fits in context.

For the financial analyst, the equivalent is the firm's market research database: historical earnings reports, industry analyses, competitive landscape studies, and valuation models—all searchable and retrievable when analyzing new opportunities.

Vector Stores: The fourth layer is the **vector store** that powers RAG—the underlying technology that makes precedent databases searchable. Rather than just keyword search (which misses synonyms and related concepts), vector stores encode documents as high-dimensional embeddings that capture semantic meaning.

When you search for "breach of fiduciary duty," the system finds not just documents containing that exact phrase but also documents about "violation of trust obligations" or "failure to act in good faith"—concepts that mean similar things even if worded differently. Each memory layer trades speed for capacity: working memory is fastest but smallest, vector stores are largest but require retrieval latency.

6.2 Retrieval-Augmented Generation

RAG enables agents to access institutional knowledge, the equivalent of asking the firm librarian "show me our best research on this issue." Traditional keyword search works but misses cases discussing the same concept using different language. Semantic search using embeddings finds conceptually similar content even when exact words differ.

The RAG Pipeline: The RAG pipeline has four steps, each transforming information to enable semantic retrieval. First, **chunking** breaks documents into semantic units while preserving metadata, so a 50-page contract becomes many retrievable segments, each maintaining reference to its source location and document context. Second, **embedding** converts each chunk into a high-dimensional vector that encodes semantic meaning, positioning similar concepts near each other in vector space regardless of the specific words used. Third, **retrieval** finds chunks similar to the query by embedding the user's question and returning chunks with similar vectors—a question about "breach of fiduciary duty" retrieves chunks discussing "violation of trust obligations" even though the exact phrase never appears. Finally, **generation** augments the agent's prompt with retrieved content, so the LLM sees both the question and relevant context from the knowledge base when formulating its response.

Advanced Patterns: The best implementations enhance basic RAG with four advanced patterns. **Hybrid retrieval** combines semantic search (embeddings) with keyword search (BM25, a standard term-frequency ranking algorithm), catching both conceptual similarity and exact term matches that pure semantic search might miss. **Query rewriting** expands ambiguous queries before retrieval, transforming vague questions like "What's the rule?" into specific queries such as "What is the legal rule governing [topic from context]?" based on conversational context. **Reranking** scores results by authority after initial retrieval, ensuring that binding precedent ranks above secondary sources even when secondary sources use more semantically similar language. **Filtered retrieval** constrains results by jurisdiction, time period, or other metadata, preventing the system from retrieving California cases when researching New York law or outdated regulations when current guidance is required.

Citation Verification: The critical requirement: never let fabricated citations reach the user. Hallucinated citations—plausible-sounding but nonexistent cases—are a known failure mode. Before any citation reaches work product, verify that the source actually appeared in retrieved context.

6.3 Domain-Specific Memory Considerations

Memory for regulated professional services requires specialized enhancements beyond generic RAG implementations. The systems must account for authority hierarchies, jurisdictional boundaries, temporal validity, and identifier normalization—requirements rarely found in consumer applications but critical for professional practice.

Authority Weighting: Not all information is equally authoritative. **Authority weighting** ensures primary authority (statutes, binding precedent) ranks higher than secondary sources. When searching for "insider trading liability," a Supreme Court opinion should outrank a law review note using more similar language.

Financial systems apply similar authority weighting: SEC no-action letters rank above law firm client alerts, official exchange rules above broker-dealer summaries, and Federal Reserve guidance above market commentary.

Jurisdiction Awareness: Jurisdiction awareness respects legal boundaries. California precedent doesn't bind Texas courts; SEC rules differ from CFTC rules. Metadata tagging during ingestion enables proper filtering. An agent researching Delaware corporate law must not surface New York case law as controlling authority, even if semantically similar.

Temporal Validity: Temporal validity matters because law changes. Citator integration validates that retrieved cases haven't been overruled. A 1985 securities case may have been good law for decades but reversed in 2023; RAG must surface the current state of the law, not historical precedent.

Financial temporal validity varies by context: milliseconds for trading data (stale prices cause losses), days for research reports (quarterly updates suffice), and effective dates for compliance rules ("What are the margin requirements *as of January 15, 2025*?").

When does memory become stale? For legal research, staleness depends on dynamism: tax law changes annually, constitutional doctrine evolves slowly. For financial data, equity prices are stale in seconds, but industry structure analysis remains valid for quarters. Effective memory systems tag temporal validity and trigger refresh when content ages beyond acceptable bounds.

Identifier Resolution: Identifier resolution normalizes citations ("123 F.3d 456" and "123 F3d 456" are the same case) and financial identifiers. Tickers change over time, and companies have multiple identifiers: CUSIP (Committee on Uniform Securities Identification Procedures numbers), ISIN (International Securities Identification Numbers), and LEI (Legal Entity Identifiers). Without normalization, retrieval fragments: half your precedent on *Smith v. Jones* does not surface because some associates cited it with spacing variations.

6.4 Matter and Client Isolation

Most critically, **matter and client isolation** prevents memory from one matter leaking into another. Law firms maintain ethical walls between conflicted representations; if an agent uses Matter A's privileged information on adverse Matter B, that's a privilege waiver and potential malpractice. Financial isolation prevents material non-public information (MNPI) exposure. An agent advising on Company X's acquisition cannot access research files containing MNPI about Company X from a separate advisory engagement.

Implement separation at the memory layer with:

Separate namespaces: Each matter gets its own isolated memory partition. Retrieval queries are scoped to the current matter's namespace, preventing cross-matter leakage.

Access controls: Role-based permissions determine which humans and agents can access which memory namespaces. Only team members assigned to Matter A can read Matter A's episodic memory or RAG results.

Audit trails: Every memory read and write is logged with timestamp, user/agent identity, matter identifier, and data accessed. Enable post-hoc compliance review and breach detection.

Secure deletion: When a matter closes or a client relationship terminates, memory must be permanently and verifiably deleted, not just logically marked inactive. Financial regulations often mandate retention schedules, but also mandate destruction after expiration.

For legal contexts, matter isolation maps to ethical walls. For financial contexts, information barriers prevent trading on MNPI or front-running client orders. Both domains treat memory isolation as a *regulatory compliance requirement*, not merely an engineering best practice.

See Section 11 for governance frameworks; the following chapter provides comprehensive treatment of memory governance controls.

6.5 Evaluating Memory Systems

How do you know if memory works? Test retrieval quality, isolation integrity, and temporal validity:

Retrieval quality: Measure precision (are retrieved documents relevant?) and recall (did retrieval find all relevant documents?). For legal research, gold-standard test sets come from known-good research memos: given the research question, does RAG retrieve the same primary authorities the human researcher cited? For financial analysis, compare retrieved earnings reports and industry studies to what an analyst would manually pull.

Isolation integrity: Verify that cross-matter queries return zero results. Matter A's agent should never retrieve Matter B's documents, even if semantically similar. Audit logs should show no unauthorized access attempts. Red-team testing deliberately attempts privilege breaches to validate controls.

Temporal validity: Track retrieval freshness. How often does the system surface overruled precedent or stale financial data? Measure lag between legal/regulatory change and knowledge base update. For high-stakes domains, daily or real-time refresh may be required.

Performance under scale: As episodic memory grows (a multi-year litigation generates thousands of documents), does retrieval latency degrade? Can the system handle concurrent queries across hundreds of active matters without contention?

6.6 From Memory to Planning

Memory provides the context agents need to plan effectively. Without memory, agents repeat failed strategies because they cannot recall what did not work. Research starts from scratch even when prior work exists. Preferences and constraints must be re-specified each session, and institutional knowledge remains inaccessible.

With memory, agents learn from experience. The agent recalls: "Last time I searched with broad terms, I got 10,000 results. This time I'll use narrower queries." Prior work product is retrieved and built upon rather than duplicated. User preferences persist across sessions, and firm expertise informs every task.

Memory enables adaptation—the “A” in the GPA+IAT framework. The agent’s behavior improves over time precisely because it 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 and execution monitoring.

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 (what facts do we need?), then dispositive motions if the law clearly favors us, settlement discussions in parallel, trial prep as a backstop. Discovery breaks into phases: initial disclosures, document requests, interrogatories, depositions. Tasks distribute across the team: senior associate handles briefing, junior associate does document review, paralegal manages scheduling and filings. Throughout, the partner monitors progress: are we on track for deadlines? Are discovery responses revealing helpful facts or should we adjust our theory?

This is **planning**: decomposing complex goals into action sequences, much like 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 know when they’re done.

Planning

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

- **Decomposition**: Breaking large tasks into manageable steps
- **Sequencing**: Ordering steps logically (what depends on what?)
- **Allocation**: Assigning steps to tools or agents
- **Monitoring**: Tracking progress toward the goal
- **Adaptation**: Adjusting the plan when circumstances change

Without planning, an agent is like an associate who keeps running searches without a research 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). The partner asks for authority that a forum selection clause is unenforceable. The associate reasons: “Key grounds are unconscionability and public policy. Start with *Atlantic Marine*.” They search, observe results, reason again: “The unconscionability cases involve consumer adhesion contracts—not our commercial situation. The public policy line is closer." They search again, refine based on results.

Each cycle has three components:

- **Thought**: Explicit reasoning about what to do next
- **Action**: Tool call to gather information or effect change
- **Observation**: 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—research questions, fact investigation, market analysis.

Plan-Execute. This pattern separates planning from execution. For document review ("Review 50 contracts for choice-of-law, forum selection, arbitration, and liquidated damages provisions"), the associate makes a plan: checklist of provisions, open each contract, record findings. Then they execute systematically. The plan does not change because the task is well-defined.

Plan-Execute fits workflows with established procedures: due diligence checklists, compliance reviews, document assembly. You create the plan upfront and execute methodically. Research variants like ReWOO (which separates reasoning from observation to reduce token usage) and LLMCompiler (which optimizes execution graphs for parallelism) enable parallel tool calling when steps are independent, though the basic 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 one agent)
- Prepare exhibits (another agent)
- Draft jury instructions (another agent)

Each specialist may decompose further. This enables parallelization and specialization, mirroring how litigation teams work with multiple associates and paralegals handling different workstreams simultaneously.

See Section 10 for detailed treatment of multi-agent coordination patterns.

7.2 Choosing the Right Planning Pattern

Selecting the right pattern depends on task structure and required autonomy level:

The autonomy column matters for governance. Higher-autonomy patterns require more sophisticated oversight:

Plan-Execute (Moderate autonomy): The agent operates within tight bounds defined by the plan. Oversight focuses on plan validation and output review.

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

ReAct (Higher autonomy): The agent makes decisions about what to search, what to pursue, when to stop. Oversight requires explicit termination mechanisms, confidence thresholds, and reasoning trace review.

Hierarchical (Distributed autonomy): Multiple agents make decisions. Oversight requires clear delegation contracts, escalation paths between agents, and coordination monitoring.

Match oversight rigor to autonomy level.

7.3 Understanding the Task Before Planning

Before planning, agents must understand what they're being asked to do. Section 3 covers intent extraction in detail. For planning purposes, the key outputs are:

Task classification: Is this exploratory (ReAct), structured (Plan-Execute), or complex (Hierarchical)?

Constraints: What bounds the work? Deadlines, budgets, scope limitations.

Success criteria: How will we know when we're done? What deliverable is expected?

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

7.4 Budget Architecture

Without explicit resource budgets, agents can run indefinitely. This is the "runaway associate" problem: you asked for two cases, the associate gives you fifty because they didn't know when the answer was sufficient.

Budget Types. Four budget types provide control over agent execution, each addressing a different dimension of resource consumption. Token budgets limit LLM API consumption, preventing expensive runaway reasoning loops where the agent keeps elaborating without making progress. Time budgets enforce deadlines by stopping execution after a fixed duration—perhaps 10 minutes—if no meaningful progress has occurred. Tool call budgets prevent runaway tool loops by capping the number of external calls; after 20 searches without progress, the agent should escalate rather than continuing to search. Cost budgets cap total spending in dollars, particularly important when using expensive models or external APIs where unconstrained execution could generate substantial charges.

These budgets cascade through levels: session budgets constrain entire engagements, task budgets allocate resources to specific work items, and subtask budgets subdivide further. A legal research task might receive a 30-minute time budget and 50,000-token limit; if it spawns subtasks, those subtasks share the parent budget rather than each receiving unlimited resources.

Cost at Scale. Token costs compound across agentic workflows. Consider a credit facility review: a 200-page document requires roughly 80,000 tokens to ingest. Each section analysis might consume 10,000–20,000 tokens across reasoning and tool calls. Retrieval from precedent databases adds tokens. Multi-iteration refinement multiplies costs.

A comprehensive review might consume 500,000–1,000,000 tokens. At illustrative pricing (late 2025: roughly $3–15 per million input tokens for leading models; verify current rates), that's $2–15 per review in API costs alone—before infrastructure, storage, or human review time.

For portfolio management running continuously, costs accumulate differently: thousands of small queries per day rather than occasional large tasks. Monitor aggregate daily/weekly costs, not just per-task.

Economic Considerations. When does agent assistance cost less than human work? Retrieval-heavy tasks (research, document review) show the clearest ROI when agents reduce hours substantially. Judgment-intensive tasks show less clear ROI when extensive human revision is required. The critical variable is human review time: agent output requiring extensive correction may cost more than human-only work.

Billing norms are evolving: some firms pass efficiency gains through as reduced hours, others add technology fees, others use fixed-fee arrangements. ABA Formal Opinion 512 requires competence regardless of tools and reasonable billing (American Bar Association Standing Committee on Ethics and Professional Responsibility 2024). Transparency about AI assistance enables clients to evaluate the value proposition.

Graceful Degradation. When budgets tighten, agents should degrade gracefully rather than failing completely. Tiered outputs provide value at every budget level: minimal budget delivers the controlling statute with citation; moderate budget adds key holdings; full budget delivers comprehensive analysis. The user receives something useful regardless of where termination occurs.

Soft limits at 75–80% of budget warn the agent that resources are running low, prompting it to prioritize completion over exploration. If the agent has found adequate authority, it should synthesize rather than searching for more. Hard limits at 100% terminate execution and return whatever partial results exist. A budget-aware agent that delivers partial results is more useful than one that fails completely, and partial results often suffice for the user's immediate needs.

7.5 Knowing When to Stop

Perhaps the most critical planning capability is knowing when to stop. Section 8 provides comprehensive treatment; for planning purposes, four categories of stopping conditions guide agent behavior.

Success conditions terminate execution when the goal is achieved: the research question is answered, the document is reviewed, the analysis is complete. The agent returns its result and stops. Resource exhaustion terminates execution when budget limits are reached, returning partial results or escalating for additional allocation. Confidence thresholds terminate execution when uncertainty is too high for autonomous action, escalating for human review rather than proceeding with unreliable conclusions. Error conditions terminate execution when repeated failures indicate a problem that retrying will not solve.

Define explicit stopping rules, just as you would instruct an associate: "If you find three on-point circuit opinions that all agree, you're done. If you've searched for two hours and found nothing, come talk to me." Agents need the same clarity about when their work is complete.

7.6 Guardrails and Loop Detection

Even with budgets and termination conditions, agents can get stuck in unproductive loops. Multiple mechanisms detect and prevent these patterns: step limits, reflection checkpoints, external watchdogs, and meta-policies. Section 8.4 provides comprehensive treatment of loop detection and guardrail mechanisms in the context of termination.

7.7 From Planning to Termination

Planning answers how agents decompose work, but every plan must end. The next two questions address the boundaries that contain autonomous execution.

Termination (Q7, Section 8) answers: how does an agent know when it is done? This involves defining success criteria so the agent can recognize completion, budget limits so the agent cannot run indefinitely, and completion recognition so the agent delivers results rather than continuing to refine. Escalation (Q8, Section 9) answers: how does an agent know when to ask for help? This involves confidence thresholds that trigger human review when uncertainty is high, authority boundaries that prevent agents from exceeding their mandate, and human-in-the-loop integration that makes escalation smooth rather than disruptive.

Without clear termination, agents run forever. Without escalation, agents exceed authority. These boundaries define the safe operating envelope for autonomous execution, ensuring that agents remain useful tools rather than becoming uncontrolled processes.

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

Every professional learns to recognize completion. The research memo is done when you've found sufficient authority and synthesized it coherently. The due diligence is done when you've reviewed all material documents and reported findings. The trade is done when the order executes and settles. Knowing when work is complete—and when it isn't—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 asked for two relevant cases, the associate gives you fifty because they did not know when enough was enough.

Termination

Termination conditions define when an agent should stop executing. Three outcomes are possible:
Success: The goal is achieved. Deliver the result.
Failure: The goal cannot be achieved. Report why and stop.
Escalation: The agent cannot determine success or failure. Transfer to human judgment.
Termination implements the "T" in the GPA+IAT framework. Without termination, agents lack the sixth property that distinguishes agentic systems from runaway processes.

8.1 Termination Condition Categories

Five categories of termination conditions bound agent execution:

Success Conditions: The most obvious termination is when the goal is achieved and the agent can return the result.

Completeness criteria: Have all required elements been produced? For document review: all provisions on the checklist have been analyzed. For research: the legal question has been answered with supporting authority. For portfolio rebalancing: allocations match targets within tolerance.

Quality thresholds: Is the output good enough? For a research memo: are conclusions supported by binding authority? For a risk assessment: have material risks been identified and analyzed? Quality thresholds often require human judgment—the agent can check completeness but may not assess quality reliably.

Convergence criteria: Has the agent stopped learning new information? If the last three searches returned no new relevant authority, the research may be saturated. If the last five portfolio adjustments produced diminishing improvement, optimization may have converged.

Resource Budgets: Hard limits prevent runaway execution by capping consumption across multiple dimensions. Token budgets stop execution after a specified threshold—say, 50,000 tokens—preventing expensive reasoning loops that would otherwise continue indefinitely. Time budgets enforce deadlines by terminating after a fixed duration, ensuring that research tasks do not consume an entire day when an hour was expected. Iteration budgets cap tool calls, stopping after twenty searches to prevent infinite loops where the agent keeps trying slightly different queries without progress. Cost budgets provide the most direct control, halting execution after spending a dollar amount (perhaps $5 in API calls) to limit financial exposure.

These budgets cascade and interact: a task might hit its time limit before exhausting its token budget, or vice versa. Budget exhaustion does not mean failure; partial results may still be valuable. But the agent must stop and report rather than continuing indefinitely.

Confidence Thresholds: Confidence thresholds gate actions on certainty, creating a decision boundary between autonomous execution and human review. When confidence is high, the agent delivers its answer. When confidence drops below a calibrated threshold—perhaps 80%—the agent stops and escalates rather than proceeding with uncertain information. This mirrors how associates should work: "I'm not confident this is right. Let me ask the partner before proceeding."

Calibrating these thresholds is challenging. Agents may be overconfident, proceeding when they should escalate, or underconfident, escalating unnecessarily and providing no value. Effective calibration requires testing against known outcomes, comparing agent confidence to actual accuracy, and adjusting thresholds until the agent escalates at appropriate uncertainty levels.

Error Conditions: Agents must recognize when things are going wrong and terminate rather than compounding errors. Repeated tool failures signal infrastructure problems: if Westlaw times out three times consecutively, the agent should stop rather than retrying indefinitely while consuming budget. Inconsistent data—a revenue figure in the 10-K that does not match the earnings release—requires human investigation, not agent guesswork about which source is correct. Constraint violations demand immediate termination: if the planned action would exceed position limits or breach confidentiality, the agent must stop before acting, not after.

Some tasks prove impossible as specified. Analysis may reveal conflicting requirements, missing prerequisites, or logical impossibilities. In these cases, the agent should report the impossibility honestly rather than proceeding with a compromised approach that satisfies the letter of the instruction while violating its spirit.

Escalation Triggers: Some situations require human judgment regardless of whether the task has succeeded, failed, or consumed its budget. Novel situations that do not match training patterns need human expertise to navigate. High-stakes decisions warrant human approval even when the agent is confident, because the consequences of error justify the overhead of review. Authority boundaries define what the agent can do autonomously; actions beyond those boundaries require escalation by design, not because something went wrong.

Section 9 provides comprehensive treatment of when and how to escalate.

8.2 Defining Success Criteria

Vague goals produce unclear termination. “Research the statute of limitations” could mean finding one relevant case or exhaustively surveying all circuits. Effective success criteria take several forms, each providing a different signal that work is complete.

Completeness checklists enumerate what must be delivered. For credit agreement review, the checklist might require identifying all financial covenants, comparing them to market terms, flagging provisions that differ from the firm’s template, and summarizing material risks. The agent terminates when all checklist items are complete. Sufficiency thresholds define “enough” without requiring exhaustive coverage. For case research, sufficiency might mean finding at least three on-point circuit opinions, or if circuits conflict, identifying the leading case from each side. The agent knows when sufficiency is reached without searching every database.

Convergence criteria recognize diminishing returns. If three consecutive searches return no new relevant authority, the research may be saturated; further searching is unlikely to yield value. Deliverable specifications define the output format—a two-page memo with executive summary, analysis, and recommendation—so the agent knows what success looks like, not just what success requires.

Consider how experienced attorneys instruct associates: “If you find clear Ninth Circuit authority, you’re done. If the circuits are split, map the split and recommend which approach applies to our facts. If you can’t find binding authority after two hours, come talk to me.” Agents need the same clarity.

8.3 Recognizing Failure

Not every task succeeds, and agents must recognize failure and report it honestly. Negative results are still results: “I searched all major databases and found no authority on point” is a valid finding that the attorney needs. The absence of authority is information. Agents should report negative results explicitly rather than continuing to search indefinitely in hope of finding something.

When failure occurs, diagnostic reporting explains what was attempted and why it failed. A useful failure report might state: “I searched Westlaw, Lexis, and Bloomberg Law using [specific queries]. Zero results suggest either the issue is novel or the search terms are wrong. Recommend manual review of secondary sources or consultation with practice group expert.” This gives the human actionable information rather than a bare “task failed” message.

Partial completion should be acknowledged honestly. If the agent completed analysis of Articles 1-4 before a tool failure, it should report what was accomplished and what remains: “Articles 5-8 remain unanalyzed.” Partial results may still be valuable, and preserving them prevents wasted effort. Where possible, root cause identification helps humans decide next steps: Was this a tool failure that will resolve itself? Impossible requirements that need rethinking? Insufficient information that requires additional input? The agent's diagnosis informs the human's response.

8.4 Guardrails and Loop Detection

Even with well-defined termination conditions, agents can get stuck in unproductive loops—searching repeatedly without progress, rephrasing queries slightly, finding nothing, rephrasing again. Multiple mechanisms detect and prevent these loops.

Step limits provide the simplest guardrail: after N steps, stop and require human approval to continue. This prevents unbounded execution regardless of what the agent thinks it is accomplishing. Progress detection monitors whether recent actions produced value: if the last five actions yielded no new information, the agent may be stuck and should trigger reflection or escalation. Reflection steps build self-assessment into the workflow, periodically asking meta-questions: "Am I making progress toward the goal? Have my recent actions been productive? Should I try a different approach or escalate?"

External watchdogs monitor agent behavior from outside the agent's own reasoning. If the same tool is called repeatedly with nearly identical parameters, an external system can recognize the loop pattern and intervene. Meta-policies encode loop detection rules directly: calling the same tool with the same parameters more than three times is probably a loop, so stop and escalate.

Without loop detection, agents will eventually get stuck in production. The question is not whether it will happen, but whether you will detect it when it does.

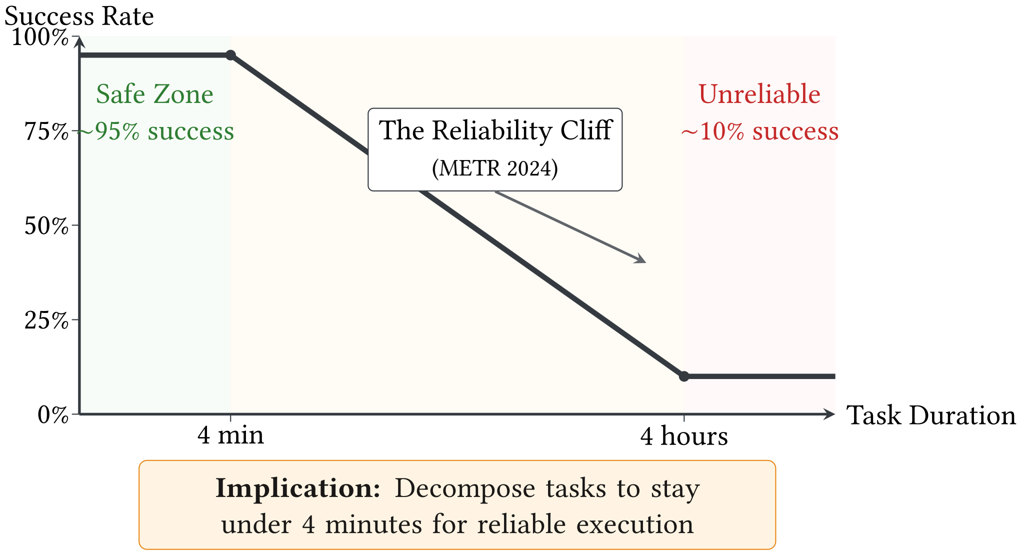
8.5 The Reliability Cliff

Independent benchmarking reveals a sharp reliability boundary. METR (Model Evaluation and Threat Research) tested agents across standardized task suites varying in duration and complexity. The results:

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—from near-100% to under-10%—defines the current boundary between reliable and unreliable agent deployment.
Implication: Decompose tasks aggressively. Keep individual agent tasks short. Insert human checkpoints between phases. Don't expect autonomous completion of multi-hour workflows.

The reliability cliff has several causes, each contributing to the dramatic drop in success rates as task duration increases. Compounding errors are perhaps the most fundamental: each step introduces error probability, and these probabilities multiply. A 95%-accurate retrieval step followed by 90%-accurate reasoning followed by 85%-accurate action yields roughly 73% end-to-end accuracy, before



Implication: Decompose tasks to stay under 4 minutes for reliable execution

Figure 5: 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.

accounting for sequencing decisions. Over a four-hour task with dozens of steps, these compounding errors accumulate into near-certain failure.

Planning fragility compounds the problem. Agents frequently select suboptimal tool sequences, get stuck in loops, or fail to recognize when their approach is not working. A human would step back and reconsider; agents often persist with failing strategies. Integration brittleness adds another failure mode: tool APIs return unexpected formats, authentication tokens expire, rate limits trigger. Each integration point is a potential failure mode, and complex tasks touch many integration points.

Design for this reality. Decompose aggressively. Validate at checkpoints. Assume agents will fail and design for graceful degradation.

8.6 Graceful Degradation

When termination occurs before full completion, agents should degrade gracefully rather than failing completely. Tiered outputs provide value at every budget level: at 20% budget, deliver the controlling statute with citation; at 60% budget, add key holdings; at 100% budget, deliver comprehensive analysis. Partial results are better than nothing, and users can decide whether partial results suffice or warrant additional investment.

Progress preservation saves intermediate state so humans can resume where the agent stopped. If the agent analyzed 30 of 50 contracts before budget exhaustion, that work should not be lost when execution terminates. Clear status reporting communicates exactly where things stand: “Completed 60%

of task. Remaining: Articles 5-8 unreviewed due to budget exhaustion. Findings so far: [summary]." The human knows what was accomplished and what remains.

Beyond reporting status, agents should recommend next steps when possible. "Recommend allocating additional 30 minutes to complete review" gives the human a concrete decision to make. "Remaining work is routine; recommend proceeding with partial findings" helps humans assess whether additional effort is worthwhile. The goal is a handoff that enables informed human decision-making, not a handoff that forces the human to start over.

8.7 Evaluating Termination Capabilities

When evaluating agent systems, assess termination capabilities against six criteria that distinguish robust systems from fragile ones.

Success criteria clarity determines whether termination is predictable. Are termination conditions explicit? Can you predict when the agent will stop? Systems with vague or implicit termination conditions produce unpredictable behavior. Budget enforcement determines whether limits actually constrain execution. Test by setting tight budgets and verifying the agent actually stops; some systems log budget exhaustion but continue anyway. Loop detection determines whether the agent recognizes when it is stuck. Test with impossible tasks or unavailable tools; a system without loop detection will spin indefinitely.

Failure reporting determines whether failures are actionable. When tasks fail, does the agent explain why? A bare "task failed" message forces the human to investigate; a detailed explanation enables informed response. Graceful degradation determines whether early termination preserves value. When stopped early, does the agent preserve partial results? Is status clearly reported? Escalation integration determines whether handoffs work smoothly. When termination requires human judgment, does the human receive sufficient context to decide? A handoff that requires the human to start from scratch is a handoff that failed.

8.8 From Termination to Escalation

Termination defines when agents stop, but not all stopping is the same. Success termination means the task is complete; deliver the results. Failure termination means the task is impossible; report why. Escalation termination means the agent cannot determine success or failure on its own; human judgment is required.

The third category is critical. An agent might complete its search and find conflicting authority, leaving it unable to determine whether the research question has been answered. It might approach a decision that exceeds its authorization. It might recognize that the situation is novel in ways that make its confidence unreliable. In each case, the right response is not to terminate with a result or a failure, but to terminate with a request for human input.

Section 9 examines this closely related question: when should an agent stop autonomous operation and ask for human help? Termination and escalation together define the boundaries of autonomous execution. Without termination, agents run forever. Without escalation, agents exceed authority. These boundaries make agent deployment safe—or at least safer.

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

The best junior associates know when to go to the supervisor. They don't interrupt the partner with every question, but they also don't 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've researched for two hours and can't 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."

Agent systems need 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 knowing where to draw 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 want from junior professionals. Agents should do the same.

9.1 When to Escalate

Three categories of triggers warrant escalation:

Mandatory Escalation Triggers: Some situations *require* human involvement regardless of the agent's confidence:

Budget exhaustion: The agent approaches resource limits (tokens, time, iterations, cost). Rather than stopping silently, escalate with a progress summary: "I've used 80% of the research budget. Here's what I found. Options: (a) grant additional budget, (b) conclude with current findings, (c) provide strategic guidance on where to focus remaining effort."

High-stakes actions: Certain actions require human approval regardless of agent confidence: filing court documents, sending client communications, executing large trades, making regulatory submissions. These are *approval gates*—the agent prepares, the human authorizes.

Authority boundaries: The action exceeds what the agent is authorized to do autonomously. Even if the agent is confident in its recommendation, organizational policy may require human sign-off above certain thresholds.

Irreversible actions: Actions that cannot be undone warrant escalation. Once you file with the court or execute the trade, you cannot take it back. Pre-approval prevents irreversible errors.

Confidence-Based Escalation: Uncertainty about the right answer or approach triggers escalation:

Low confidence on output: The agent's uncertainty exceeds acceptable thresholds. For legal research: "I found conflicting circuit authority. I'm not confident which rule applies in our jurisdiction." For portfolio analysis: "Correlations have spiked beyond historical norms—model assumptions may be violated."

Conflicting information: Data sources disagree. The 10-K revenue does not match the earnings release. Two authoritative sources give different answers. Human judgment is needed to resolve the conflict.

Novel situations: The scenario does not match patterns the agent has seen. Novel legal questions, unusual market conditions, unprecedented fact patterns—these warrant human expertise.

Ambiguous instructions: Despite clarification attempts (Section 3), the agent remains uncertain about what is being asked. Rather than guessing, escalate for clarification.

Error and Anomaly Detection: Something has gone wrong and the agent cannot fix it:

Repeated tool failures: If Westlaw times out three times consecutively, escalate: "Research tool unavailable. Options: (a) wait and retry, (b) use alternative platform, (c) proceed with manual research."

Data anomalies: Red flags in the data warrant human investigation. Revenue figures that do not reconcile, filing dates that seem wrong, parties that appear on multiple sides of a transaction.

Constraint violations: The task as specified would violate a policy or constraint. "Executing this trade would exceed the position limit. Please confirm override or adjust the order."

Impossible requirements: Analysis reveals the task cannot be completed as specified. Conflicting requirements, missing prerequisites, logical impossibilities. Report the problem rather than proceeding with a compromised approach.

9.2 How to Escalate

Effective escalation provides the human with everything needed to make a decision:

Five-Part Escalation Structure:

1. Situation summary: What task is the agent working on? Brief context for a human who may not have been following closely.

2. Progress to date: What has been accomplished? What remains? Do not make the human start from scratch.

3. Escalation trigger: Why is the agent escalating now? What specific condition triggered the handoff?

4. Information gathered: What relevant information has the agent found? Even if incomplete, partial findings are valuable.

5. Options or recommendations: What are the possible paths forward? If the agent has a recommendation, state it with supporting 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. Found conflicting circuit authority on when discovery is triggered.
>
> **Trigger**: Low confidence. The Ninth Circuit and Second Circuit 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 more conservative test and note the split; (b) research district court authority in our jurisdiction; (c) seek partner guidance on which test likely applies.
>
> **Recommendation**: Option (c): this appears to be a fact-intensive question where partner judgment on the strength of our facts would be valuable.

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 exceeds single-approver threshold ($500K total).
>
> **Findings**: Recommended trades would realize $45K in short-term gains and $12K in losses. Net tax impact: approximately $8K additional liability.
>
> **Options**: (a) Approve full trade list; (b) modify to prioritize tax-loss positions; (c) execute in tranches over multiple days.
>
> **Recommendation**: Option (a) preferred if reducing exposure is urgent; Option (b) if tax optimization is priority.

9.3 Human-in-the-Loop Patterns

Five patterns integrate human oversight into agent workflows, each suited to different risk profiles and organizational needs.

Approval Gates: The agent prepares work product; the human authorizes execution. Essential for irreversible or high-stakes actions. A litigation agent drafts the court filing, presents it for review, receives approval, then submits. The agent handles preparation; the human controls execution.

Checkpoint Reviews: Human verification at milestones prevents error propagation. A research agent completes legal research, presents authorities, receives confirmation of direction, then proceeds to drafting. Each checkpoint catches misalignment before significant effort is wasted.

Confidence-Based Escalation: High confidence triggers autonomous execution; low confidence triggers escalation. A compliance agent processes clear-pass cases automatically while escalating ambiguous situations. Balances efficiency with safety.

Human-as-Tool: The agent invokes human expertise like any other tool. When encountering questions beyond its capabilities, it queries the relevant expert, incorporates the response, and proceeds. Human expertise becomes a resource invoked when needed, not a bottleneck for all decisions.

Reversibility Classification: Oversight level tracks action consequences. Fully reversible actions (drafts, research) proceed autonomously. Partially reversible actions (communications) receive checkpoint review. Irreversible actions (filings, trades) require pre-approval. Oversight proportional to risk.

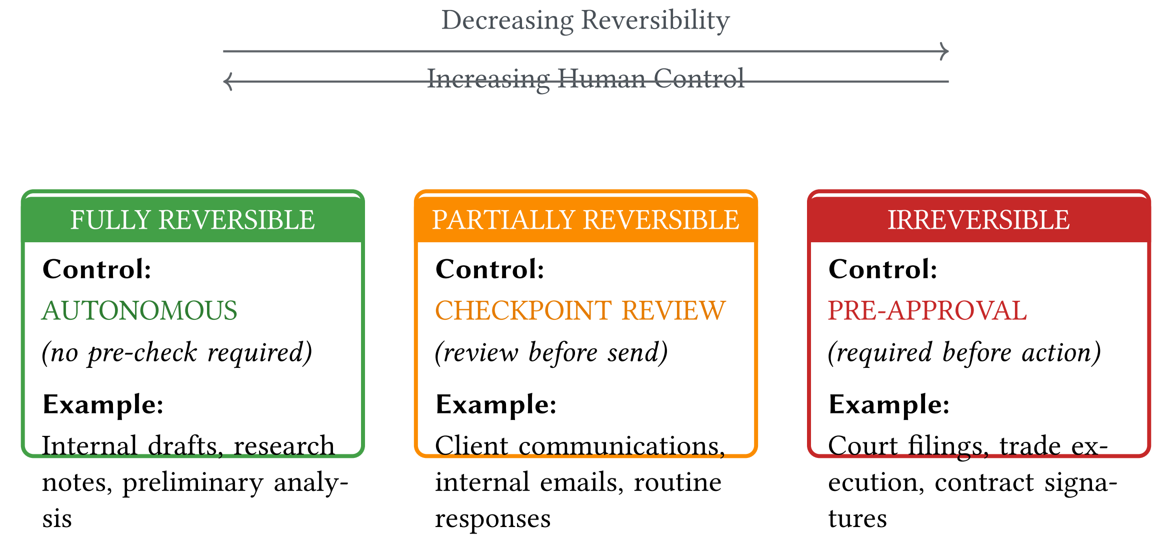


Figure 6: Oversight spectrum showing how the reversibility of agent actions determines the level of human control required. Fully reversible actions may proceed autonomously, partially reversible actions require checkpoint review, and irreversible actions demand pre-approval before execution.

9.4 Domain-Specific Escalation Requirements

Legal Practice. Legal practice imposes domain-specific escalation requirements rooted in professional responsibility rules. ABA Model Rule 1.1 requires competent representation, which means agents must escalate when matters exceed their training or the supervising attorney's oversight capacity. Novel legal questions, complex strategic decisions, and matters outside the firm's expertise all warrant escalation—not because the agent failed, but because competence requires recognizing the limits of one's capabilities.

Privilege protection demands particular vigilance. Attorney-client privilege can be waived by disclosure, and any action that might expose privileged information to third parties requires attorney review before execution. Similarly, conflicts of interest require human judgment because agents cannot assess conflicts without access to the full picture of firm relationships; potential conflict situations must escalate to conflicts counsel. Candor to tribunal creates another mandatory escalation trigger: if the agent discovers authority adverse to the client's position that the attorney may have an obligation to disclose, it must escalate immediately rather than suppressing the finding.

Financial Services. Financial services escalation requirements arise from regulatory obligations and fiduciary duties. Suitability and fiduciary duty require that investment recommendations be suitable for the client, but agents cannot fully assess suitability without understanding client circumstances, risk tolerance, and investment objectives. Recommendations therefore escalate for adviser review before reaching the client.

Regulatory thresholds create mandatory escalation points. Large trades trigger reporting requirements; position accumulations have disclosure thresholds; certain transactions require pre-approval. Agents must recognize when thresholds approach and escalate for compliance review before crossing them. Material non-public information demands immediate escalation: if the agent encounters potential MNPI, it cannot assess whether the information is material or public, so compliance judgment is required. Risk limits similarly require human judgment. Trading agents must escalate when proposed actions would breach risk limits, even if the trade itself appears profitable; risk management requires human judgment about whether limit exceptions are appropriate.

9.5 Evaluating Escalation Mechanisms

When evaluating agent systems, assess escalation mechanisms against six criteria that distinguish effective systems from those that either escalate too much or too little.

Coverage determines whether all appropriate situations trigger escalation. Test with edge cases: novel situations that should clearly require human judgment, conflicting data that the agent cannot resolve, near-threshold conditions that might slip through. A system with coverage gaps will occasionally proceed autonomously when it should not. Calibration determines whether thresholds are set appropriately. Too sensitive and the agent escalates everything, providing no value; too loose and it proceeds when it should not. Calibrate thresholds against real scenarios, adjusting until the agent escalates when practitioners agree it should.

Latency determines how quickly escalation reaches the right human. For urgent matters—a margin call, a filing deadline, a client emergency—escalation must be immediate. For routine matters, queued escalation may suffice. Routing determines whether escalation reaches the right person. Complex legal questions should reach senior attorneys, not paralegals; risk limit breaches should reach risk managers, not operations staff. Misrouted escalation wastes time and may produce inadequate responses.

Context quality determines whether the human can actually decide. Test by reviewing escalation messages and asking: could you make a decision from this information alone? If the human must investigate further before responding, the escalation is incomplete. Response handling determines whether the agent correctly incorporates human guidance. Test the full cycle, not just escalation initiation; an agent that escalates well but ignores responses provides only the illusion of human oversight.

9.6 From Escalation to Delegation

Escalation moves control *up*: from agent to human supervisor. But agents can also move control *sideways* by delegating subtasks to other agents. Where escalation says "I need human help," delegation says "I need specialist help."

Section 10 examines the next question: how does an agent work with other agents? Delegation patterns enable complex workflows where multiple specialized agents collaborate, each with its own escalation paths back to human oversight. A coordinating agent might delegate research to a legal research specialist, analysis to a financial modeling specialist, and drafting to a document generation specialist—while each specialist retains the ability to escalate to humans when it reaches its own limits.

The combination of escalation (vertical) and delegation (horizontal) defines the full topology of human-agent collaboration. Escalation ensures human oversight. Delegation enables specialization and scale. Together, they make complex agentic workflows possible while maintaining the human control that regulated professions require.

10 How Does an Agent Work with Other Agents?

Complex matters require coordination. The M&A partner does not do everything personally but instead coordinates specialists: corporate counsel reviews governance documents, tax specialists analyze structure, antitrust counsel assesses regulatory risk, employment lawyers review executive agreements. Each specialist has deep expertise in their domain. The partner's job is orchestration: defining what each specialist should produce, ensuring deliverables integrate coherently, and synthesizing conclusions for the client.

A portfolio manager coordinates similarly: research analysts provide company-specific analysis, traders handle execution, risk managers monitor exposure, compliance officers verify regulatory adherence. Complex trades require all these perspectives; no single person has all the expertise.

Agent 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 in narrow domains, orchestrators that coordinate their work, 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:

Specialization: A securities law agent can be optimized for SEC regulations, loaded with relevant precedent, and equipped with EDGAR tools. A separate tax agent handles tax implications with different tools and knowledge. Neither needs to be expert in the other's domain.

Security isolation: Each agent gets minimum necessary permissions. The research agent can read legal databases but cannot file documents. The filing agent can submit to CM/ECF but cannot access client financial data. If one agent is compromised, damage is contained.

Parallel execution: Independent workstreams proceed simultaneously. The document review agent analyzes contracts while the research agent investigates legal issues. Neither waits for the other.

Vendor diversity: Different agents can use different models or providers. Use a specialized legal model for research, a general model for drafting, a fast model for classification. Multi-agent enables best-of-breed selection.

Scale management: Context windows have limits. Rather than cramming everything into one context, decompose across agents, each with focused context.

The tradeoffs: coordination overhead (agents must communicate), debugging complexity (failures span multiple agents), and additional security surface (more agents means more potential attack vectors).

10.2 Agent-to-Agent Protocol (A2A)

The Agent-to-Agent Protocol standardizes how agents collaborate, complementing MCP's tool integration. If MCP is how agents access resources, A2A is how agents delegate work to specialists.

A2A Concepts: A2A uses familiar professional concepts:

Agent Cards: Capability statements, like a specialist's résumé listing expertise, input requirements, and output formats. Before delegating, the coordinator reviews the Agent Card to confirm the specialist can handle the task.

Tasks: Units of delegated work, like engagement letters specifying scope, constraints, and deadlines. The coordinator creates a Task; the specialist accepts and executes.

Artifacts: Work products returned upon completion. Draft memos, analysis reports, structured data. The specialist produces Artifacts; the coordinator integrates them.

Communication Channels: Support for asynchronous, long-running work. You assign research Monday; the memo arrives Friday. Channels enable status updates and clarification requests during execution.

Task Lifecycle: Agent collaboration follows five phases mirroring professional delegation:

A2A Task Delegation

1. DISCOVERY: Find specialist via Agent Card → *Like finding co-counsel through a directory*
2. DELEGATION: Create Task with goals, constraints, deadline → *Like an engagement letter*
3. EXECUTION: Specialist works independently, may request clarification → *Like an associate researching*
4. DELIVERY: Specialist returns Artifacts → *Like submitting a draft memo*
5. COMPLETION: Coordinator reviews, approves, or requests revision → *Like partner review*
Key insight: A2A enables delegation without micromanagement: you define WHAT, the specialist decides HOW.

10.3 Multi-Agent Patterns

Three patterns organize multi-agent collaboration:

Sequential Delegation: Specialists work in sequence, each building on prior work: Coordinator → Research Agent → Analysis Agent → Drafting Agent → Coordinator synthesizes. Simple but slow—each specialist waits for prior completion.

Parallel Delegation: Independent specialists work simultaneously: Securities, Tax, and Employment Agents analyze an acquisition concurrently; Coordinator integrates findings. Faster, but requires task independence.

Hierarchical Delegation: Specialists delegate to sub-specialists: Lead Due Diligence Agent delegates to Document Review and Legal Research agents, each of which may further delegate. Handles complex matters but introduces coordination overhead and cascading failure risk.

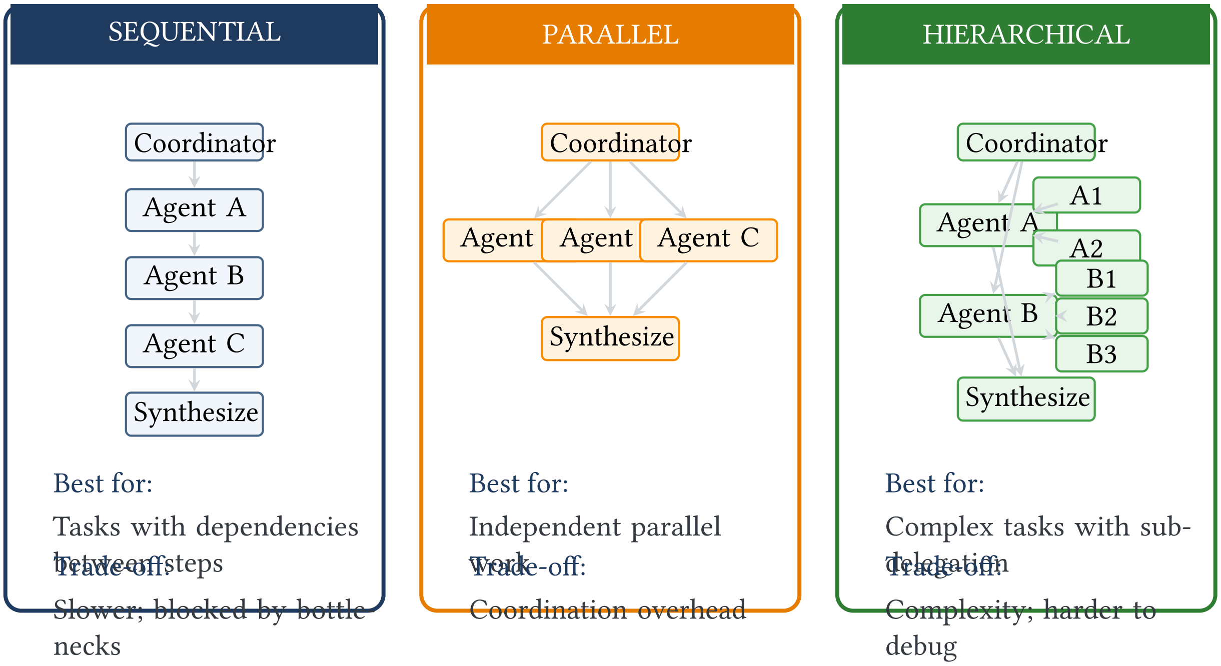


Figure 7: Three multi-agent orchestration patterns. Sequential delegation chains agents in order, ideal for dependent tasks but vulnerable to bottlenecks. Parallel delegation runs agents concurrently, maximizing throughput for independent work but requiring coordination. Hierarchical delegation enables sub-agents to handle specialized sub-tasks, providing flexibility for complex workflows at the cost of debugging complexity.

Dual Protocol Strategy: Production systems typically require both MCP and A2A working in concert:

MCP: Agent-to-tool communication. Each specialist agent uses MCP to access its domain-specific tools (databases, calculators, external systems).

A2A: Agent-to-agent coordination. The orchestrator uses A2A to delegate tasks to specialists and receive artifacts.

Example: M&A Due Diligence: The orchestrator receives "Conduct due diligence on Target Company acquisition." It delegates via A2A:

Document Processing Agent: Indexes the data room, classifies documents, extracts key terms. Uses MCP to access document management systems and OCR tools.

Financial Analysis Agent: Analyzes financial statements, builds models, identifies risks. Uses MCP to access financial databases and calculation tools.

Legal Risk Agent: Reviews contracts for problematic provisions, researches legal issues. Uses MCP to access legal research platforms and precedent databases.

Each specialist returns Artifacts via A2A. The orchestrator synthesizes into a comprehensive due diligence report.

Throughout: MCP handles tool access; A2A handles coordination. Neither protocol alone suffices.

10.4 Multi-Agent Workflow Examples

Legal: Regulatory Compliance Assessment: A fintech company asks: "What regulatory approvals do we need to launch this product?"

Orchestrator: Receives query, decomposes into regulatory domains.

Securities Agent: Analyzes whether the product involves securities. Uses MCP to search SEC guidance, no-action letters, case law. Returns: "Product likely constitutes a security under Howey. Registration or exemption required."

Banking Agent: Analyzes banking law implications. Uses MCP to search OCC, FDIC, state banking guidance. Returns: "No bank charter required, but state money transmitter licenses may apply."

Consumer Protection Agent: Analyzes CFPB jurisdiction and state consumer laws. Returns: "UDAP exposure; recommend clear disclosures and complaint handling procedures."

AML Agent: Analyzes Bank Secrecy Act obligations. Returns: "FinCEN registration required; implement KYC/AML program."

Orchestrator: Synthesizes into comprehensive regulatory roadmap with prioritized action items.

Financial: Large Block Trade Execution: A portfolio manager requests: "Execute a $50M block trade in XYZ Corp, minimizing market impact."

Orchestrator: Decomposes into analysis and execution phases.

Market Agent: Analyzes current liquidity, trading patterns, optimal execution windows. Uses MCP for market data feeds. Returns: "Average daily volume $200M. Recommend VWAP execution over 2 days."

Compliance Agent: Verifies trade doesn't breach limits or create reporting obligations. Uses MCP for compliance databases. Returns: "Trade within limits. 13F amendment will be required at quarter-end."

Risk Agent: Assesses impact on portfolio risk metrics. Uses MCP for risk engine. Returns: "Trade increases sector concentration by 2%. Within policy limits."

Execution Agent: Implements the execution strategy. Uses MCP for order management systems. Returns: "Order placed. Will report execution quality upon completion."

Orchestrator: Monitors execution, coordinates any adjustments, reports completion to portfolio manager.

10.5 Multi-Agent Risks

Multi-agent systems introduce failure modes and security challenges beyond single-agent problems:

Coordination Failure Patterns:

Deadlock: Agents wait for each other cyclically; neither proceeds. Prevention: clear task dependencies, timeouts, circular-wait detection.

Divergent conclusions: Specialists reach incompatible results. The orchestrator must detect conflicts and either reconcile or escalate.

Cascading errors: Incorrect output from one agent propagates through dependent agents. Prevention: validate inputs at each handoff rather than trusting upstream agents.

Coordination overhead: Communication consumes tokens, time, and cost. For simple tasks, overhead may exceed specialization benefits. Use multi-agent when complexity justifies it.

Accountability gaps: When something fails, which agent is responsible? Prevention: comprehensive logging at every delegation, clear audit trails, defined accountability per subtask.

Security Controls: Multi-agent systems require security controls at the coordination layer:

Agent identity: Each agent must have verifiable identity. Cryptographic signatures authenticate Agent Cards and task responses. Impersonation attacks must be prevented.

Authorization controls: Not every agent can delegate to every specialist. Access control policies define valid delegation relationships.

Information barriers: Legal conflicts and financial Chinese walls must be enforced across agent boundaries. The agent working on Company A's acquisition cannot delegate to agents with access to Company A's confidential information from other engagements.

Audit trails: Every delegation must be logged: who delegated what to whom, when, with what constraints, and what was returned. Enable post-hoc analysis and compliance review.

Task validation: Specialists should validate that tasks fall within their authorized scope. Reject tasks that would require accessing forbidden data or taking unauthorized actions.

10.6 Protocol Selection Guidance

Use MCP when: You need immediate tool access—database queries, document retrieval, calculations.

Use A2A when: You need to delegate work that requires judgment, iteration, or extended execution time.

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

Use both when: Complex workflows combine tool access (each specialist uses MCP) with coordination (orchestrator uses A2A).

Maturity (late 2025): MCP is production-ready for tool integration. A2A is maturing—stable spec, active pilots, but cross-vendor reliability remains uneven. Design fallbacks to human coordination where A2A would ideally apply.

10.7 From Delegation to Governance

Delegation distributes work across agents, creating governance challenges single-agent systems avoid. Accountability becomes complex: when a multi-agent workflow errs, responsibility could lie with the coordinator, the specialist, or the human who approved output. Information barriers applying to human teams must apply to their agents—an agent cannot access data its human principal could not. Audit trails must span the entire delegation tree; when regulators ask what happened, the trail must show every delegation, handoff, and decision point.

Section 11 previews governance requirements; the following chapter provides comprehensive treatment including accountability models, barrier enforcement, and audit architecture.

11 How Do We Keep the Agent Safe?

Every professional organization has compliance programs, audit functions, and oversight structures. The law firm has conflicts committees, billing review, and quality control. The financial institution has risk management, compliance monitoring, and internal audit. These functions do not do the work themselves; they ensure the work is done safely, ethically, and in compliance with applicable rules.

Agent 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 governance requirements. Every architectural choice enables or constrains oversight. This section previews governance across all ten questions; the following chapter provides comprehensive treatment.

Agent Governance

Agent governance encompasses the policies, controls, and oversight mechanisms that ensure agents operate safely, ethically, and in compliance with applicable requirements. Governance spans the agent lifecycle: design, deployment, operation, and retirement.

Governance is not optional for regulated professional services. 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 throughout 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, when opposing counsel demands production of the agent's reasoning, when a client questions why the agent recommended a particular strategy—architecture determines whether you can answer.

Professional duties are non-delegable: attorneys remain liable for AI-assisted work product, and fiduciaries remain accountable for AI-informed recommendations. The following chapter details those obligations. This chapter provides the architecture to meet them. Section 13.1 provides a comprehensive mapping of architectural choices to governance implications.

11.2 Governance Requirements by Question

Each of the ten questions creates specific governance requirements:

11.3 Security Essentials

Five security controls are essential for any agent deployment in regulated contexts:

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 constraint violations
3. **Least privilege**: Grant minimum necessary tool access to limit the scope and impact of failures

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

4. **Audit logging**: Maintain comprehensive action logs for accountability and investigation
5. **Matter/client isolation**: Enforce confidentiality boundaries to protect privileged and confidential information

These controls map to the ten-question framework:

- Input separation protects Q2 (Intent) from manipulation
- Output validation governs Q4 (Action)
- Least privilege limits Q3 (Perception) and Q4 (Action)
- Audit logging enables Q7 (Termination) review and Q8 (Escalation) tracking
- Matter/client isolation enforces Q5 (Memory) boundaries

11.4 Transparency and Explainability

Regulators and clients increasingly require explanations for agent decisions. Four levels of transparency serve different audiences, illustrated here with a breach detection agent:

Level 0 (Output only): Just the answer—"Suspicious transaction flagged." Sufficient for routine, low-stakes queries where the consumer trusts the system.

Level 1 (Summary with sources): Conclusion plus citations—"Transaction #45921 flagged; exceeds threshold in Rule 203(b)(1)." Enables verification without full reasoning.

Level 2 (Reasoning outline): Key analytical steps plus sources—"Flagged because: (1) $150K exceeds $100K threshold, (2) counterparty on watchlist, (3) timing matches known pattern." Appropriate for substantive work product requiring review.

Level 3 (Execution trace): Structured record of tool calls, retrieved documents, and decision points—full database queries, rule evaluation steps, and confidence scores. Enables audit and debugging.

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

11.5 Auditability vs. Retention

A tension exists between comprehensive logging (for audit) and data minimization (for privacy and compliance). The resolution is *not* "log everything forever." Instead:

Structured logging: Log structured decisions, not raw chain-of-thought. Structure enables selective retention.

Tiered retention: Short-term operational logs (full detail, days to weeks); medium-term audit logs (structured decisions, months to years); long-term compliance archives (minimal but sufficient, as required).

Redaction at capture: Apply privacy and confidentiality filters before logging, not after.

Legal hold integration: Retention schedules must yield to preservation obligations when litigation is anticipated.

The following chapter provides detailed retention frameworks for legal and financial contexts.

11.6 Forward to Chapter 8

This chapter answered *how to build an agent*. The ten questions (Table 5) provide architectural foundations; each creates governance requirements that the architecture 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 (autonomy, persistence, goal dynamics), accountability structures, regulatory compliance frameworks, and worked examples in legal, financial, and audit contexts.

Architecture provides the foundation; governance provides the controls. Together, they enable responsible deployment of agentic systems in regulated professional services.

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 agents 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

The Scenario: 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.

Ten Questions Applied: The framework guides every design choice in this architecture.

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

Q2 (Intent): The agent extracts intent: document review task, borrower perspective, focus on material deviations from market and template. 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 facility agreement (precedent database), and market terms data (external database). Read-only access; no modifications.

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

Q5 (Memory): Episodic memory tracks analysis progress (which sections reviewed, what issues identified). RAG provides access to prior credit agreement memos and deal histories. 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. Systematic execution through financial covenants, events of default, representations, conditions precedent.

Q7 (Termination): Success criteria: all material sections reviewed, issues identified and categorized, 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 with other client matters, issues that might affect deal viability. Human-as-tool pattern for partner input on materiality thresholds.

Q9 (Delegation): Single-agent architecture for this task. Multi-agent would be appropriate if combined with separate research (legal issues) or financial modeling (covenant analysis) 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.

Workflow: The agent proceeds systematically through the review process:

1. Partner assigns matter; agent receives trigger
2. Agent retrieves credit agreement, template, market terms
3. Agent creates review plan: 15 sections, estimated analysis per section
4. Agent analyzes Section 1 (Definitions): identifies unusual defined terms, compares to template
5. Agent continues through financial covenants: flags leverage ratio that differs from market
6. Agent reviews events of default: identifies cross-default threshold below typical market terms

7. ... [continues through all sections]
8. Agent compiles findings into issues list and draft memo
9. Agent presents to associate for review
10. Associate reviews, adds context, escalates significant issues to partner

Failure Modes: Even well-designed systems fail. Understanding failure modes is crucial:

Nuanced definitions: The agent may miss that a defined term has been subtly modified from the template in ways that affect covenant calculations. Human review catches: "EBITDA is defined to exclude one-time charges, but the add-back is capped—that's unusual and limits flexibility."

Cross-document dependencies: The credit agreement references schedules and exhibits. If the agent doesn't trace these references and analyze the schedules, material issues may be missed.

Market context: "Market terms" vary by borrower credit quality, industry, and timing. The agent compares to a template, but the template may not reflect current market for this borrower's profile.

Omissions: The most dangerous failures are things the agent doesn't flag because it doesn't recognize their significance. Human expertise identifies: "There's no limitation on amendments to subordinated debt—that's a significant gap."

Mitigation: Checkpoint review after initial scan. Associate reviews agent output, not just for correctness but for completeness. Partner review of final work product. The agent accelerates but doesn't replace human judgment.

12.2 Case Study: Equity Portfolio Management

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

This is a continuous monitoring and advisory task, not a one-time analysis. The goal is to augment the PM's capacity, not to trade autonomously.

Ten Questions Applied: The framework shapes this more complex, multi-agent architecture.

Q1 (Triggers): Multiple trigger types: market data feeds (price movements, earnings releases), scheduled jobs (daily compliance check, weekly rebalancing analysis), human prompts (PM requests analysis), escalation events (position approaching limits).

Q2 (Intent): Intent varies by trigger. Price alert: assess significance and recommend action. Scheduled rebalance: generate trade list if allocations drift beyond thresholds. PM query: answer specific question about position or strategy.

Q3 (Perception): MCP Resources access market data (prices, fundamentals), portfolio data (positions, P&L), research (analyst reports, news), and compliance data (client guidelines, regulatory limits). Read-only access to trading systems.

Q4 (Action): Agents can generate recommendations and create reports. Trade execution requires PM approval—the execution action tool is behind an approval gate. No autonomous trading.

Q5 (Memory): Episodic memory tracks recent analysis, PM decisions, and rationales. RAG accesses investment research archive. Client isolation ensures each client's portfolio data is segregated.

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

Q7 (Termination): Varies by task. Monitoring: continuous (no termination). Analysis: complete when question answered. Rebalancing: complete when trade list generated and approved.

Q8 (Escalation): Escalate on: positions approaching limits, unusual market conditions, conflicting signals, any trade recommendation (PM approval required). Risk management escalation path for limit breaches.

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

Q10 (Governance): Comprehensive audit trail. Fiduciary duty documentation (rationale for recommendations). Compliance monitoring. MNPI controls (no access to deal team information).

Multi-Agent Workflow: Multiple agents coordinate to generate and validate recommendations:

1. Monitoring Agent detects: "Tech sector up 3% today; portfolio tech allocation now 35% vs. 25% target"
2. Monitoring Agent triggers Rebalancing Agent
3. Rebalancing Agent queries current positions (MCP)
4. Rebalancing Agent generates trade options: sell $20M tech, options include [specific positions]
5. Rebalancing Agent queries Compliance Agent: "Check proposed trades against guidelines"
6. Compliance Agent confirms: trades within limits, no restricted securities
7. Rebalancing Agent queries Research Agent: "Any recent negative research on proposed sales?"
8. Research Agent returns: "No material negative research; one position has earnings next week"
9. Rebalancing Agent adjusts: defer one sale until after earnings
10. Rebalancing Agent presents recommendation to PM Agent
11. PM Agent formats for human review, highlights key considerations
12. Human PM reviews, approves (or modifies), authorizes execution
13. Execution Agent (with PM approval) places orders via OMS

Multi-Agent Failure Modes: Coordination introduces additional failure vectors beyond single-agent systems.

Cascading errors: Monitoring Agent misinterprets data; Rebalancing Agent acts on bad signal; trades recommended that shouldn't be. *Mitigation*: Validation at each handoff; sanity checks on data before acting.

Coordination overhead: Communication between agents consumes tokens and time. For simple decisions, overhead may exceed value. *Assessment*: Monitor coordination costs; simplify when overhead dominates.

Debugging complexity: When recommendations are wrong, tracing the error through multiple agents is difficult. *Mitigation*: Comprehensive logging; clear attribution at each step; replay capability.

Agent disagreement: Research Agent sees positive signal; Risk Agent sees negative. How is conflict resolved? *Design*: Clear escalation when agents conflict; human resolves material disagreements.

12.3 Synthesis: Principles Across Domains

Both case studies illustrate common principles that apply across legal and financial agent deployments:

Decomposition is essential: Neither workflow attempts end-to-end autonomous completion. The credit facility review breaks into 15 section-by-section analyses rather than attempting one pass; the portfolio rebalancing separates constraint checking from trade generation from execution. Tasks are decomposed into manageable steps with human checkpoints.

Human-in-the-loop is the norm: Both architectures assume human oversight at critical junctures: attorney review of legal analysis before client delivery, PM approval of trade recommendations before execution. Agents augment professional judgment; they do not replace it.

Memory enables continuity: Both rely on episodic memory (what happened in this session) and RAG (institutional knowledge). The legal agent retrieves prior firm precedent on similar provisions; the financial agent accesses historical rebalancing decisions. Without memory, every interaction starts fresh.

Isolation is non-negotiable: Matter isolation (legal) and client isolation (financial) are architectural requirements, not optional features. The agent working on Company A's financing cannot access Company B's confidential terms, even if both are firm clients.

Failure modes are predictable: The same failure patterns appear in both domains: nuanced judgment that exceeds current capabilities, omissions where agents miss non-obvious issues, and cascading errors where early mistakes propagate through analysis. Design for these failures explicitly.

Governance is pervasive: Every architectural choice has governance implications. Audit trails document what the agent accessed and concluded; approval gates ensure human review before irreversible actions; escalation paths route uncertainty to appropriate decision-makers.

Current limitations are real: Neither architecture claims autonomous completion of multi-hour tasks. The reliability cliff constrains what agents can do today; design accordingly.

12.4 Framework Completion Checklist

Before deploying any agent 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 properly controlled? Is provenance tracked?
- □ **Action**: What can the agent do? Are irreversible actions gated? Is rollback possible?
- □ **Memory**: What persists across sessions? Is isolation enforced? What are retention policies?
- □ **Planning**: What pattern applies? Are budgets enforced? Is there loop detection?
- □ **Termination**: How does the agent know when it's done? What are success criteria? How does it handle failure?
- □ **Escalation**: When does the agent ask for help? Who receives escalations? Is context sufficient?
- □ **Delegation**: Does it coordinate with other agents? Are protocols standardized? Are barriers enforced?
- □ **Governance**: Are security controls implemented? Is there audit capability? Are professional duties met?

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

13 Conclusion: From Architecture to Governance

AI agents are organized like professional teams. Just as a law firm needs infrastructure—library access, case files, project management, supervision, escalation paths—agents need architecture. The ten questions in Table 1 map these professional structures to technical capabilities: how work arrives (Q1), how instructions become goals (Q2), how information is gathered (Q3) and actions executed (Q4), how context persists (Q5) and work decomposes (Q6), how completion is recognized (Q7) and help requested (Q8), how specialists coordinate (Q9), and how safety is ensured (Q10).

The organizational analogy is a design principle, not merely a metaphor. Agent architecture mirrors professional organization because both solve the same problems: distributing cognitive work, maintaining context, coordinating specialists, ensuring quality, and enabling oversight.

13.1 What This Lets You Do

Evaluate vendor claims critically: When a vendor says their agent "handles legal research," you know to ask: What triggers initiate research? How does it understand the research question? What tools provide information? How does it know when research is complete? What escalation paths exist? The ten questions provide an evaluation framework.

Participate in procurement decisions: You can assess whether a proposed agent system meets your organization's requirements. Does it enforce matter isolation? Does it maintain audit trails? Does it integrate with your approval workflows? You have vocabulary to specify requirements.

Design governance that maps to architecture: You understand that governance is not separate from architecture; it is enabled by architecture. Audit logging is an architectural choice that enables compliance review. Approval gates are architectural choices that enable human oversight. You can design systems where governance is built in, not bolted on.

Communicate with technical teams: You can describe what you need in terms developers understand. "I need perception tools for these three databases, action tools behind approval gates for these two operations, escalation triggers when confidence drops below threshold." Shared vocabulary enables collaboration.

Architecture Enables Governance

Every architectural choice has governance implications:

- Trigger logging enables audit of what initiated agent action
- Intent extraction enables review of what the agent understood
- Perception controls enable data governance
- Action gates enable approval workflows
- Memory isolation enables confidentiality protection
- Planning budgets enable bounded operation
- Termination criteria enable completion verification
- Escalation paths enable human oversight
- Delegation contracts enable accountability

You cannot govern what you did not architect.

13.2 Current Limitations

Honest assessment of current capabilities: agents exhibit a reliability cliff (Section 8.5) with near-perfect success on tasks under 4 minutes but under 10% on tasks over 4 hours; they excel at retrieval and systematic execution but struggle with nuanced judgment and novel situations; production systems fail unpredictably due to API changes, authentication expiration, and format variations; and multi-step workflows compound error probabilities at each step. These limitations are not permanent, but they are real today. Design systems that deliver value despite limitations, not systems that assume limitations do not exist.

13.3 Essential Resources

This chapter introduced ten fundamental questions that every agent designer must answer. Essential resources for moving from concepts to deployment:

Security: Implement the five foundational controls detailed in Section 11.3 (input separation, output validation, least privilege, audit logging, matter/client isolation) before any production deployment. The OWASP LLM Top 10 provides vulnerability taxonomy for language model applications; the NIST AI Risk Management Framework offers lifecycle guidance organized into four functions (Govern, Map, Measure, Manage) that align with enterprise risk management practices.

Protocols and Standards: The Model Context Protocol (MCP) standardizes agent-to-tool communication (Section 4, Section 5) and is production-ready as of late 2025 with thousands of available servers. The Agent-to-Agent Protocol (A2A) standardizes agent-to-agent coordination (Section 10) and is maturing under the Linux Foundation; suitable for internal multi-agent coordination, though cross-vendor interoperability is still emerging.

Research Foundations: For deeper theoretical grounding, see Xi et al. (2023) on agent architecture and design patterns (Xi et al. 2023), Yao et al. (2022) on the ReAct reasoning-action loop (Yao et al. 2022), and Park et al. (2023) on memory architecture (Park et al. 2023). For evaluation, use LegalBench (162 legal reasoning tasks) (Guha et al. 2023) and VLAIR (legal AI performance against lawyer baselines) (Bommarito et al. 2025).

Learning Paths: *Legal professionals* should start with narrowly scoped pilots focused on evaluation criteria (accuracy, audit trails, fail-safe behaviors) and validate outputs against their own analysis. *Financial professionals* should begin with read-only monitoring tasks, integrating with existing systems (Bloomberg terminals, portfolio management, compliance databases) before introducing advisory workflows. *Technical practitioners* should build a simple research agent using a framework (LangChain, LlamaIndex, CrewAI), add memory and evaluation, then build an MCP server for a real data source with monitoring and audit logging. *For everyone*: implement security controls from the beginning—retrofitting is expensive.

Staying Current: Technology advances quickly, regulation is emerging, and security risks evolve continuously. Monitor protocol specifications on GitHub, follow research venues (NeurIPS, ICML, ACL), track framework release notes. Legal practitioners should monitor ABA ethics opinions (particularly Formal Opinion 512 on supervision (American Bar Association Standing Committee on Ethics and Professional Responsibility 2024)); financial practitioners should monitor SEC guidance, FINRA communications, and prudential regulators' guidance on model risk management. Subscribe to OWASP LLM Top 10 updates and follow security researchers specializing in language model vulnerabilities.

Temporal Warning

Resources accurate as of late 2025 may not reflect subsequent developments. Protocol specifications evolve, regulatory frameworks develop, security vulnerabilities emerge. Verify currency of all technical and regulatory references before relying on them for production deployment decisions.

13.4 From Architecture to Governance

This chapter answered: *How do you build an agent?*

The ten questions provide architectural foundations. Each question maps to implementation choices. Each implementation choice enables—or forecloses—governance options.

The next chapter answers: *How do you govern an agent?*

Where this chapter focused on *capability*—what agents can do—the next focuses on *control*—ensuring agents do what they should, and only what they should. The five-layer governance stack (legal, model, system, process, culture) provides the framework. Dimensional controls (autonomy, persistence, goal dynamics) calibrate oversight. Implementation patterns translate principles into practice.

You understand agent architecture. Now you can govern it.

You cannot govern what you do not understand.

This chapter has provided that understanding.

References

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).

Bommarito, Michael J., Daniel Martin Katz, and Eric M. Detterman (2025). *VLAIR: Validating Lawyer AI Reasoning*. Benchmark comparing legal AI performance against lawyer baselines across substantive legal tasks. URL: `https://arxiv.org/abs/2503.00000` (visited on 12/04/2025).

Guha, Neel, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Alex Chohlas-Wood, Austin Peters, Brandon Walber, Nika Haghtalab, et al. (2023). “LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models”. In: *arXiv preprint arXiv:2308.11462*. 162 tasks from 40 contributors covering six types of legal reasoning; developed by Stanford and HazyResearch.

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).

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`.

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.

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.