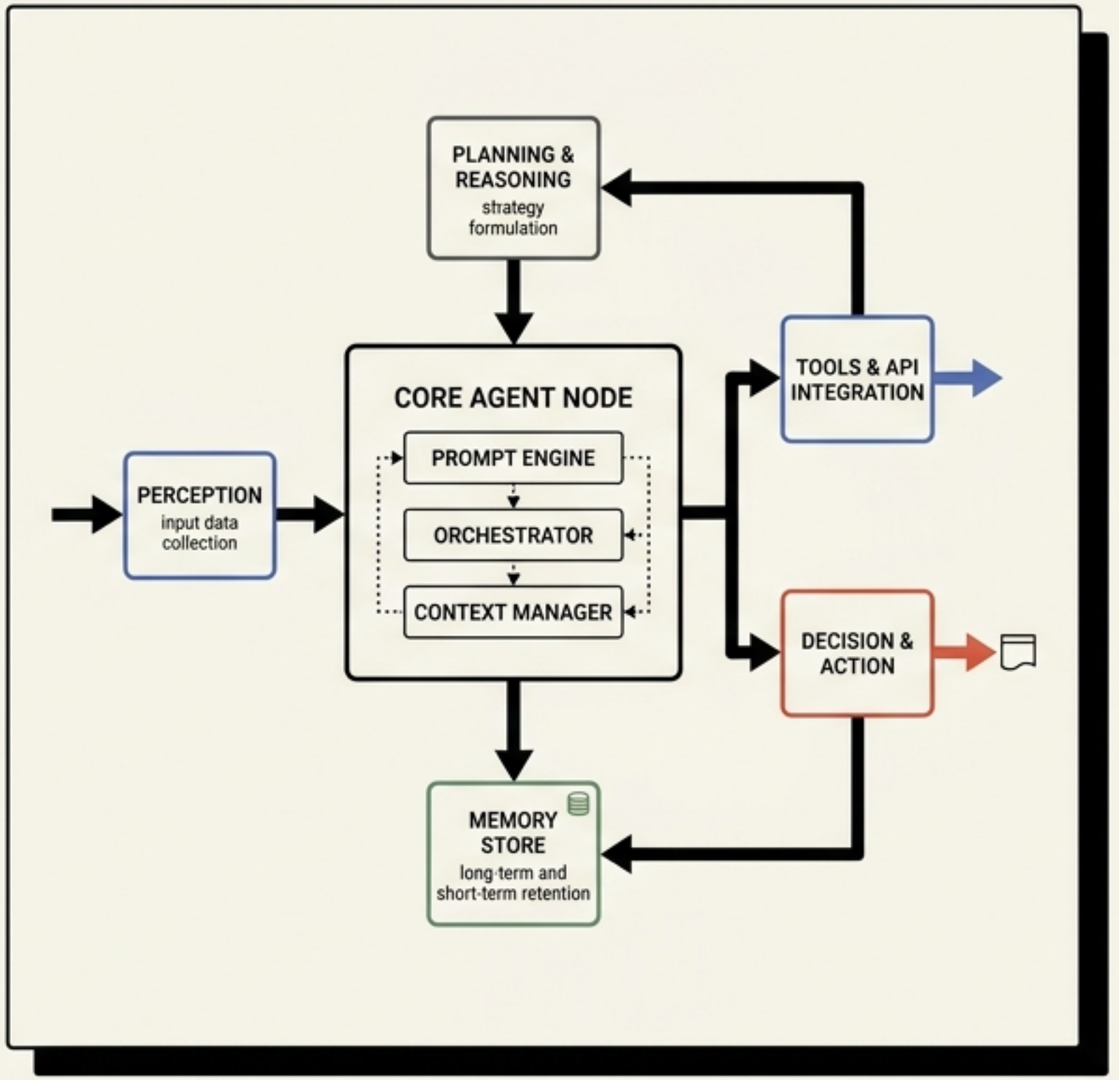
The Agentic AI Blueprint

Architecting Autonomous Systems: From Prompting to Production

---

A technical playbook for product managers and software architects moving beyond reactive LLMs.



The AI Spectrum: Evolution of Architecture

Single LLM (Reactive)

Process:

Characteristics:
Stateless processing, direct request-response, no memory.

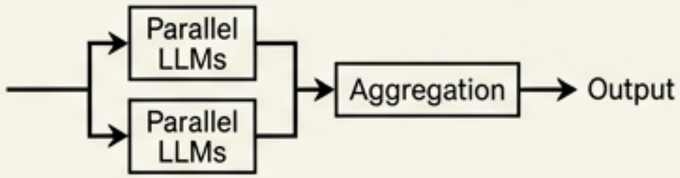
Best For:
Simple, defined tasks (summarization, sentiment).

Pros:
Fast and low cost.

Cons:
Lacks context and adaptability.

Structured Workflows (Orchestrated)

Process:



Characteristics:
Deterministic execution, explicit control flow, predefined tool chains.

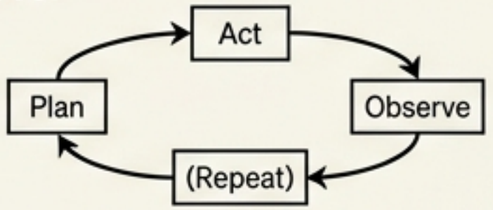
Best For:
Repetitive, multi-step tasks (document pipelines).

Pros:
Predictable and auditable.

Cons:
Rigid, unable to handle ambiguity.

Autonomous Agents (Agentic)

Process:



Characteristics:
Dynamic planning, contextual awareness, tool orchestration.

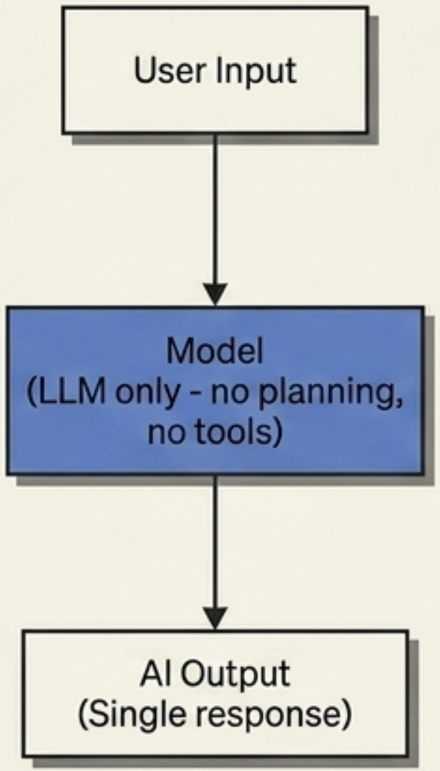
Best For:
Complex, open-ended tasks (research, troubleshooting).

Pros:
Highly flexible, learns from feedback.

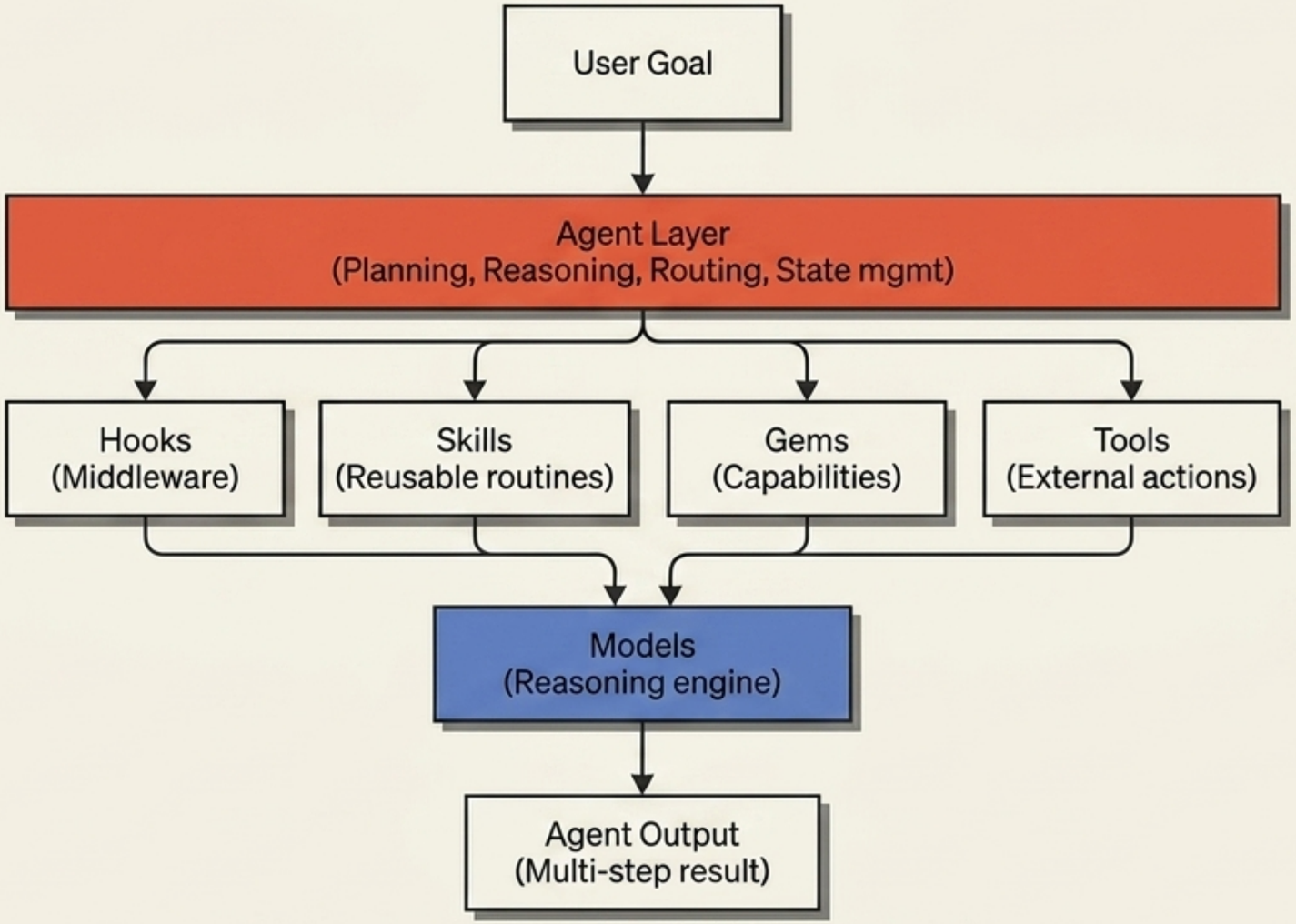
Cons:
Complex, costlier, unpredictable.

Architectural Shift: Non-Agentic vs. Agentic

Non-Agentic AI (Reactive)

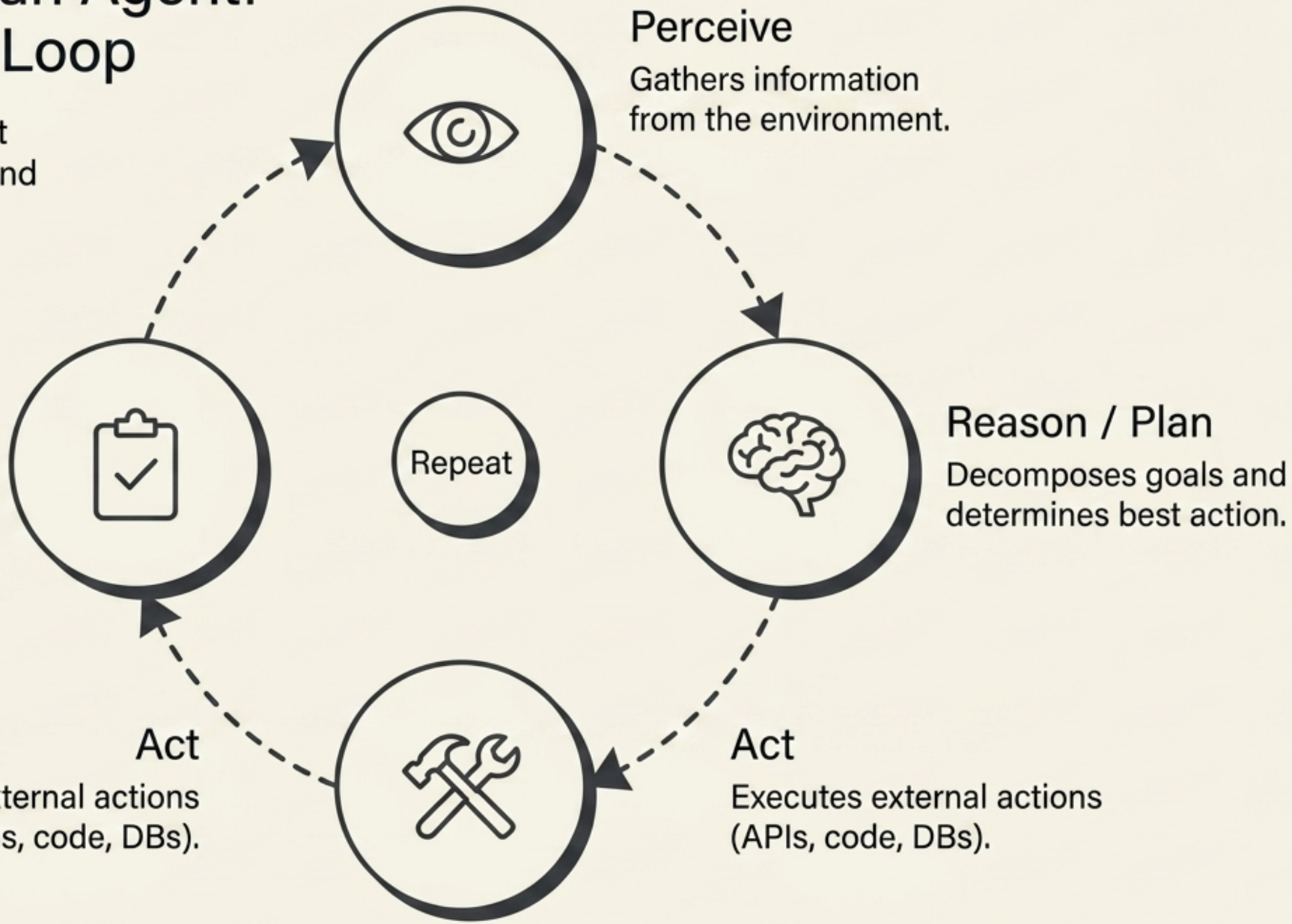


Agentic AI (Autonomous)

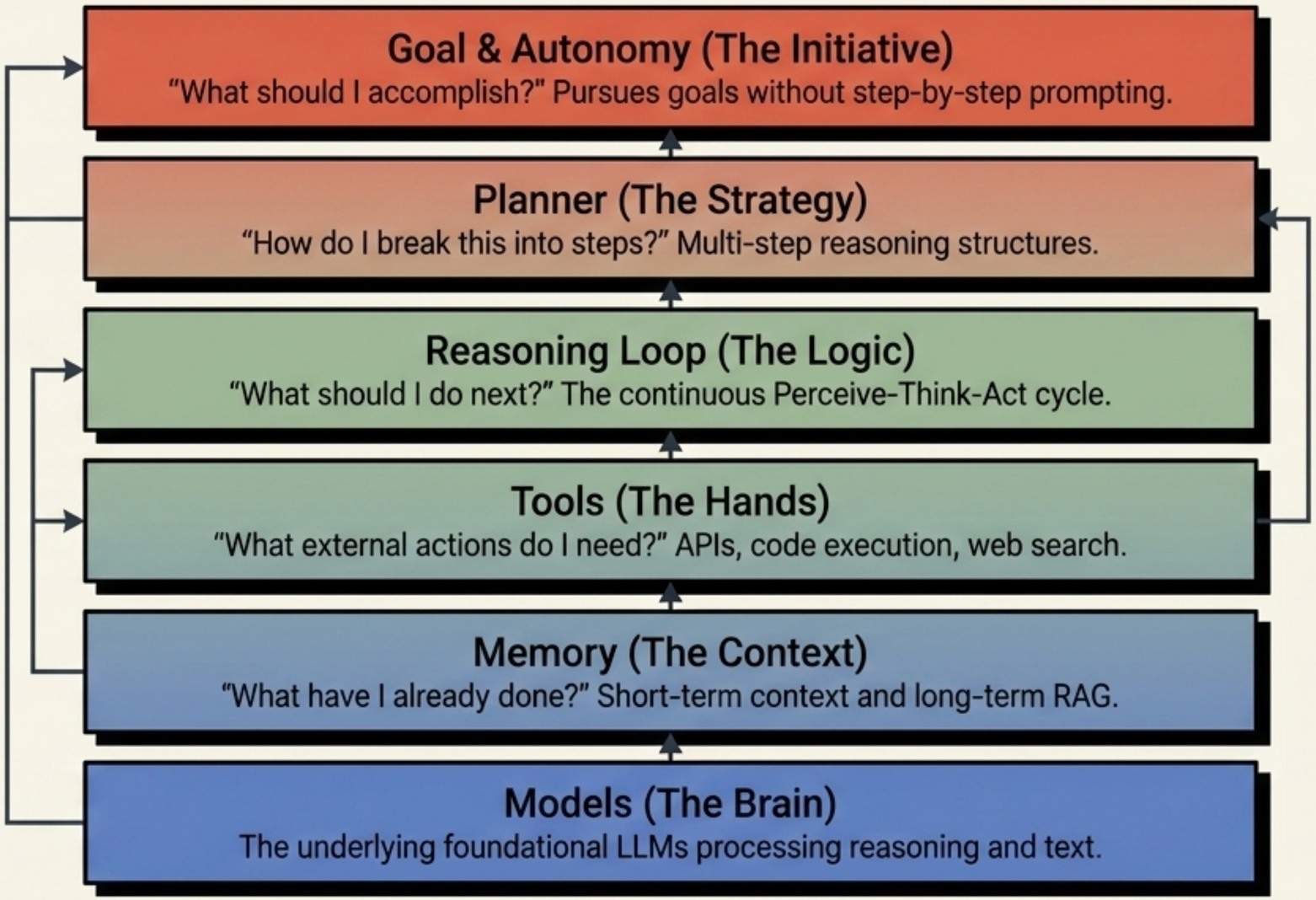


The Anatomy of an Agent: The Continuous Loop

An AI system that doesn't just respond—it acts, evaluates, and repeats until the goal is met.



The Agentic Stack: 6 Components of Autonomy



Equipping the Agent: The Developer's Toolkit

Tools (External Actions)

The AI's hands. External capabilities the AI calls to perform actions outside the model.

- File system access, HTTP requests, DB queries, Shell commands.

Gems (Micro-Capabilities)

LEGO bricks for AI systems. Small, modular, callable capabilities providing deterministic behavior.

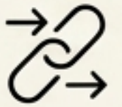
- Summarization, code execution, web search, data extraction.

Skills (Domain Routines)

Repeatable patterns of behavior. Reusable, domain-specific abilities the AI can invoke.

- “Summarize legal text”, “Rewrite code”, “Extract JSON”.

Hooks (Middleware)

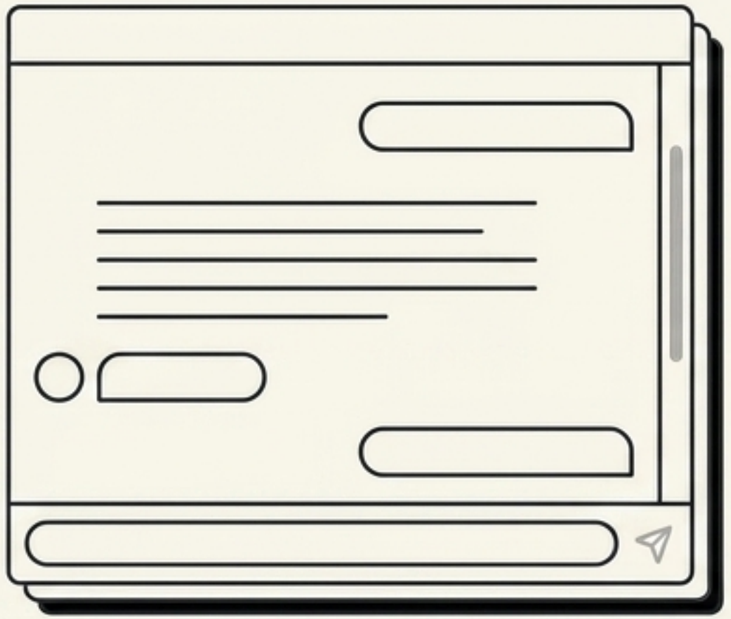


Event-based triggers to intercept or modify AI behavior dynamically.

- Pre-prompt hooks, Post-response validation, Tool-invocation overrides.

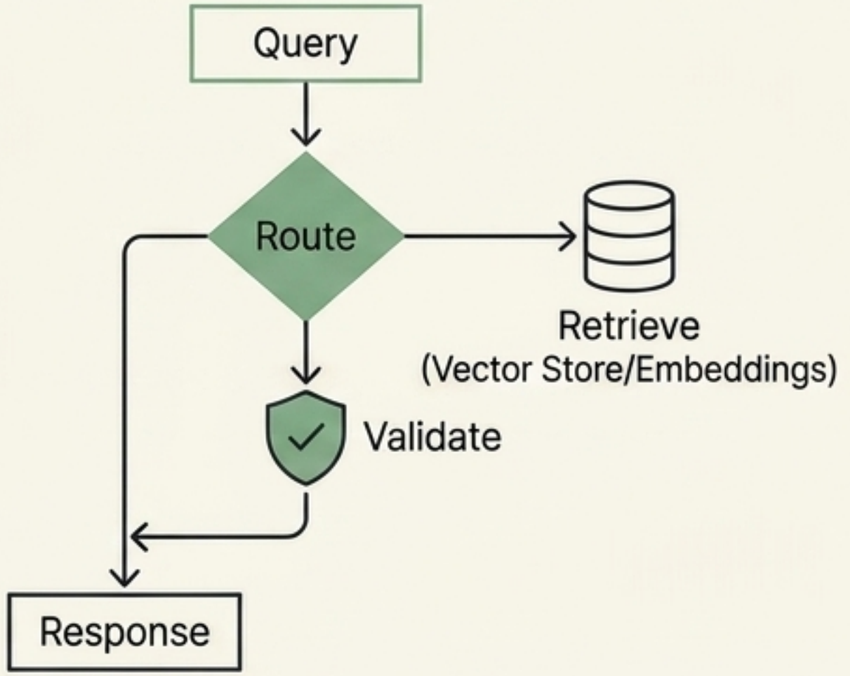
The Memory Engine: Maintaining State and Context

Short-Term Memory



The AI's working memory. Stateful interactions where the agent remembers previous messages, tool results, and partial outputs within the current session's token limits.

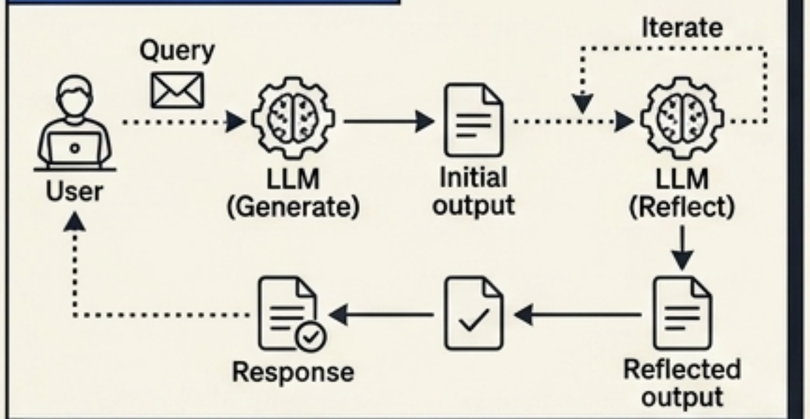
Long-Term Memory (Agentic RAG)



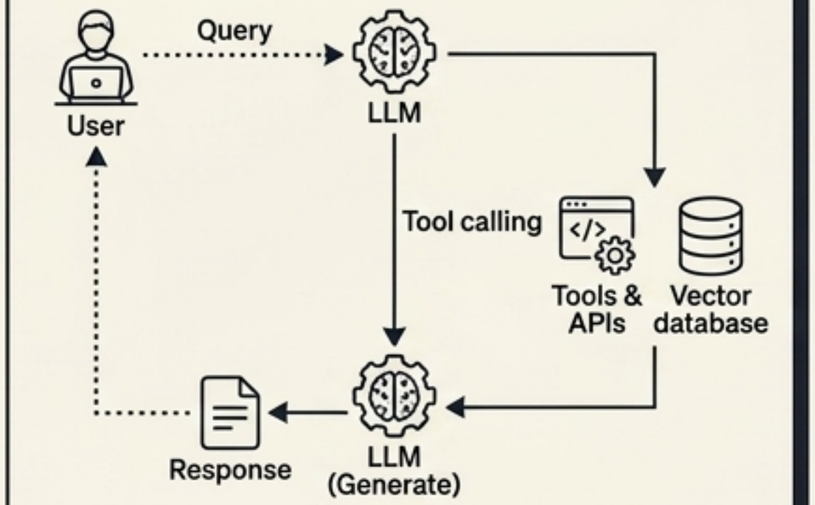
The AI's long-term memory. Uses Vector Stores (FAISS) and embeddings. The agent dynamically decides when to route queries to validate retrieved information.

The 5 Core Agentic Design Patterns

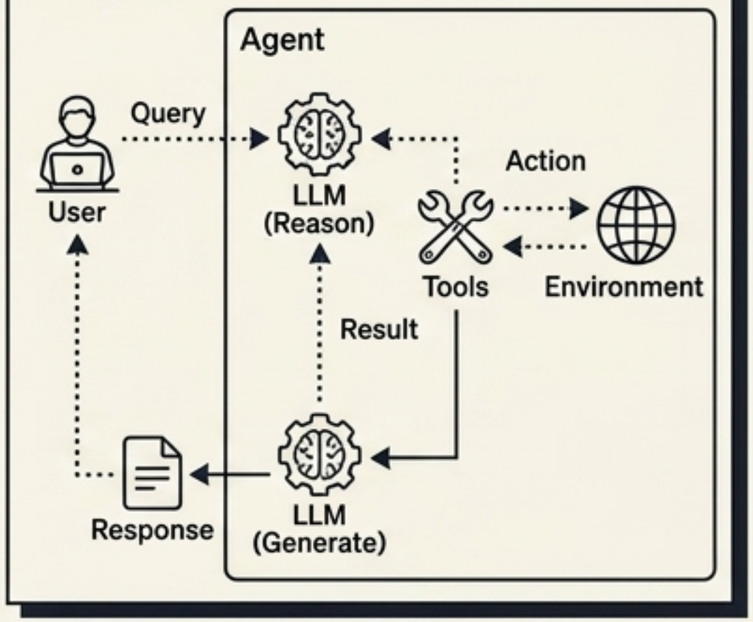
1) Reflection Pattern



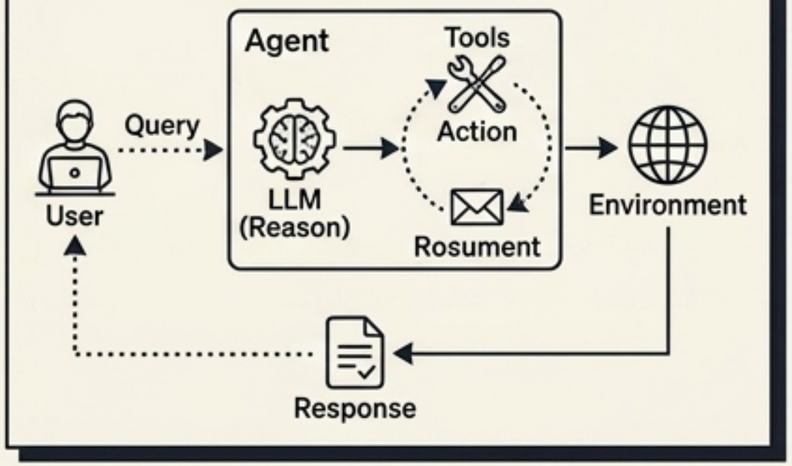
2) Tool Use Pattern



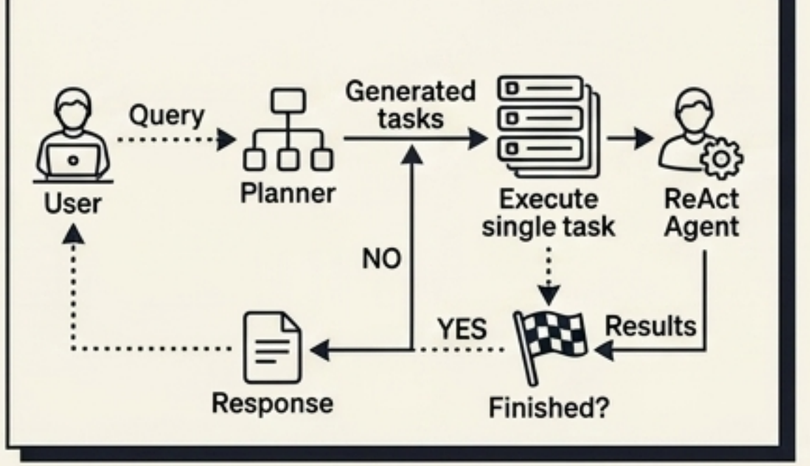
3) ReAct Pattern



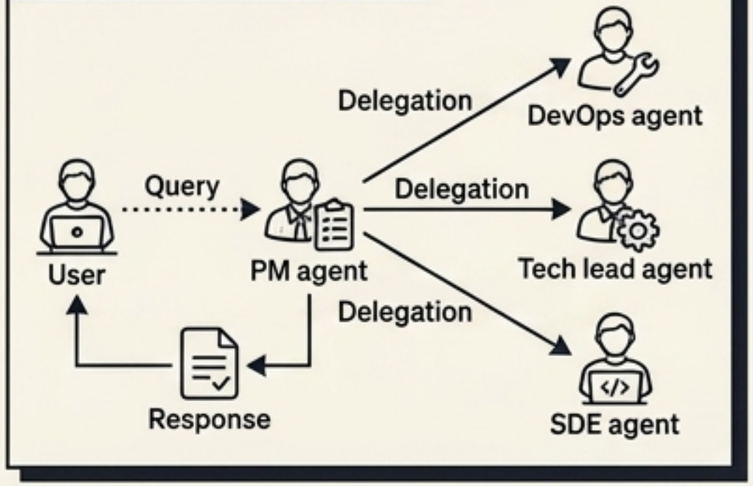
3) Planning Pattern



4) Planning Pattern

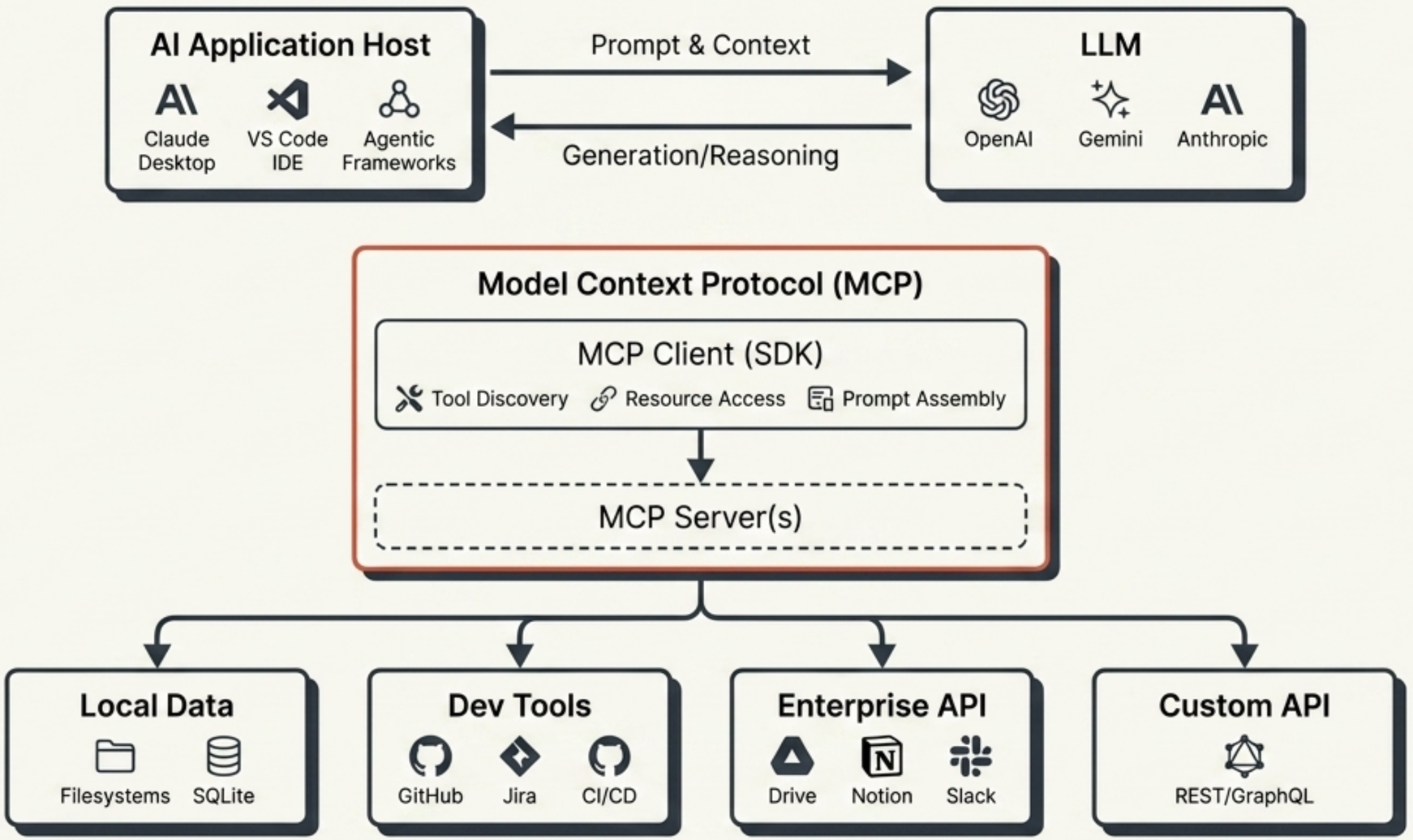


5) Multi-Agent Pattern



The Universal Connector: Model Context Protocol (MCP)

The "USB-C for AI" — a universal standard replacing custom, fragile integrations.

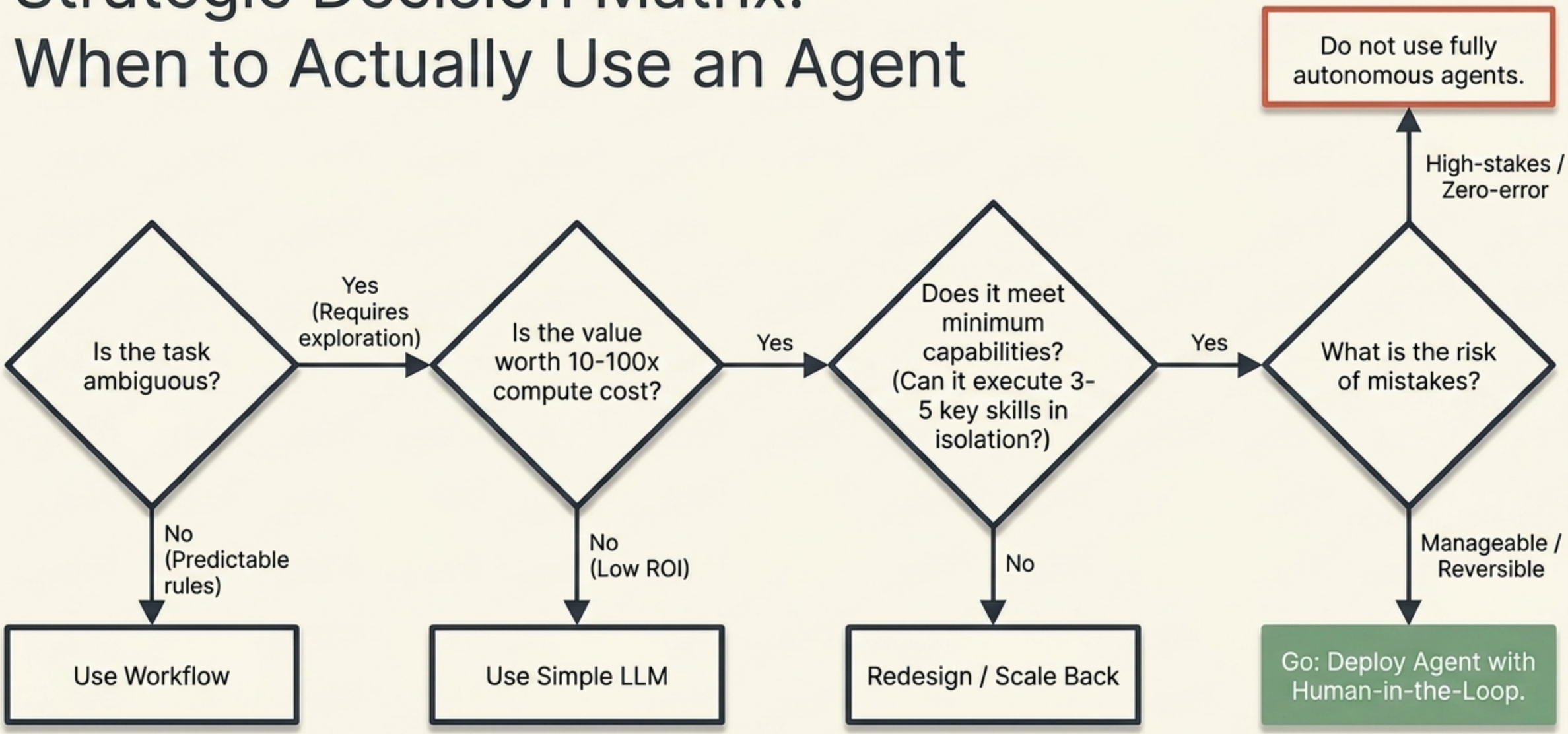


Programming the Agent: The BRIDGE Prompting Framework

Designed for complex strategic planning and agentic instruction.

B	**Background / Context** Provide the backstory or business context so the AI understands why the task is being performed.
R	**Role / Persona** Define the exact perspective or expertise the AI should adopt.
I	**Instructions / Task** State the core action the agent must execute.
D	**Details / Constraints** Specify formatting rules, limits, or strict structural requirements (Schemas).
G	**Guidance / Guardrails** Outline rules of engagement—what to prioritize and what to explicitly avoid.
E	**Examples / Evaluation** Include few-shot examples of perfect output or strict evaluation criteria.

Strategic Decision Matrix: When to Actually Use an Agent



Build Strategy: From Scratch vs. Frameworks

Building From Scratch (Custom Python)

Architecture:

Standalone classes with explicitly coded logic for perception, routing, and memory.

Pros:

Complete control over APIs and thresholds; highly transparent; excellent for debugging.

Cons:

High complexity; time-intensive; scaling requires massive boilerplate code.

Using a Framework (LangChain, CrewAI)

Architecture:

High-level abstractions handling perception, reasoning, and memory internally.

Pros:

Rapid development; built-in orchestration and router modules; scales naturally.

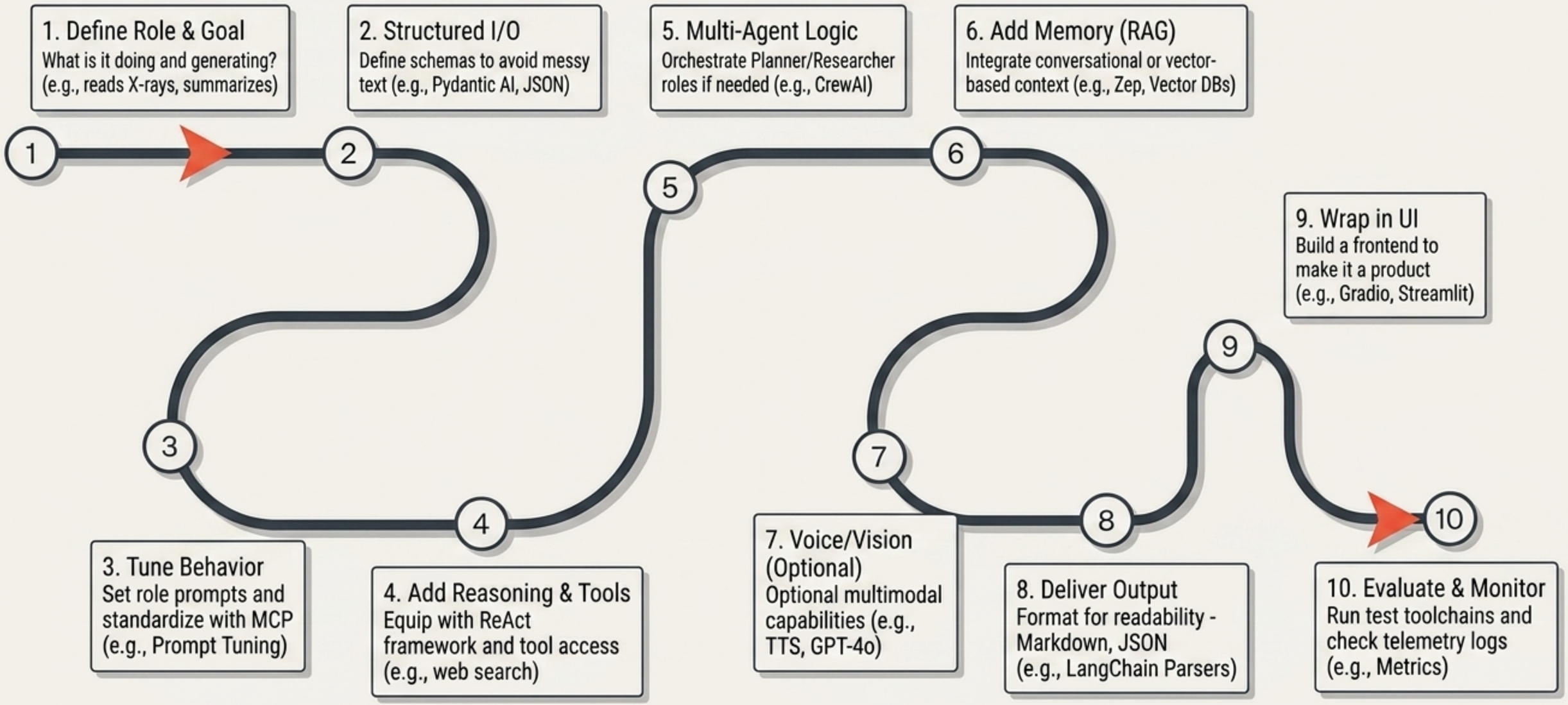
Cons:

Flexibility bounded by framework design; debugging can become a 'black box'.

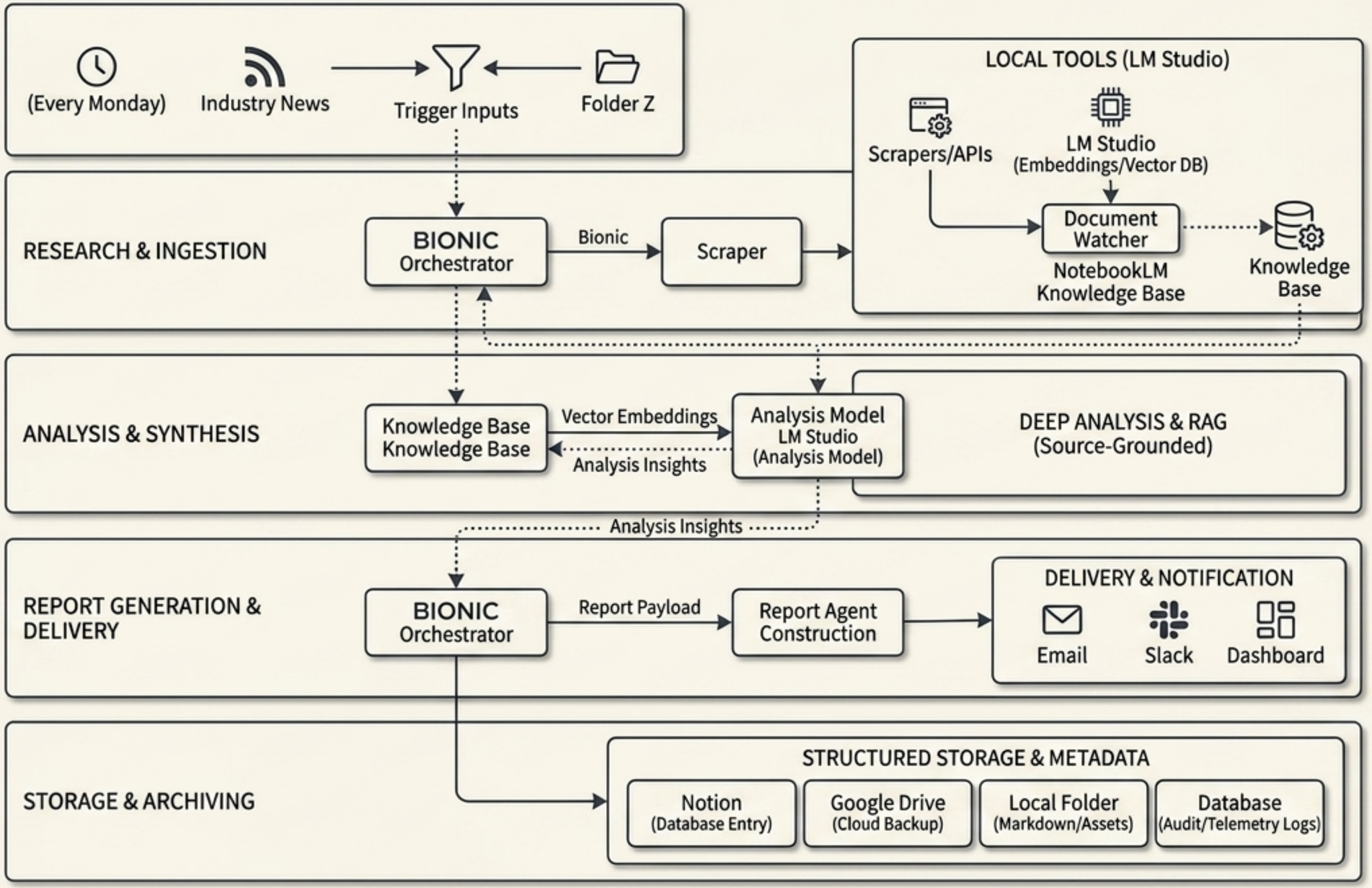
Trade-off slider

The Builder's Path: 10 Steps to Create an AI Agent

16:9 slide in 'The Architectural Whitepaper'



Anatomy of an Agentic Workflow in Production



Production Realities: Risk, Governance, & Oversight

The Human-in-the-Loop

You are always the final judge. High-stakes tasks require human review and automated validation layers.

Agents augment; they do not replace critical judgment.

Safe Deployment Phasing

- Phase 1 (PoC): Low-risk, reversible tasks. Read-only tool access.
- Phase 2 (Pilot): Moderate-risk under supervision. Approvals for critical steps.
- Phase 3 (Production): Scale with comprehensive logging (MCP, LangSmith) and observability.

Mitigating Hallucinations & Drift

Continually monitor reward functions.

Anchor the agent firmly in factual external data using RAG.

Define strict schemas to constrain output boundaries.

Agentic AI is a co-pilot with initiative. Architect the system to leverage its autonomy while protecting the boundaries of your enterprise.