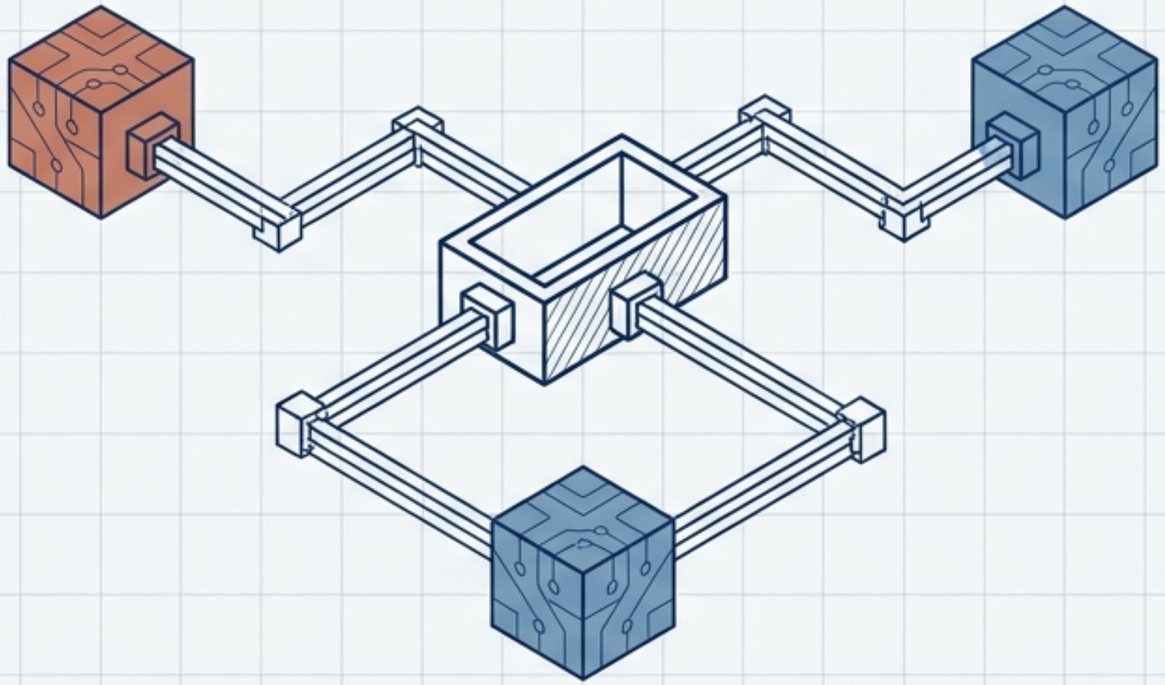
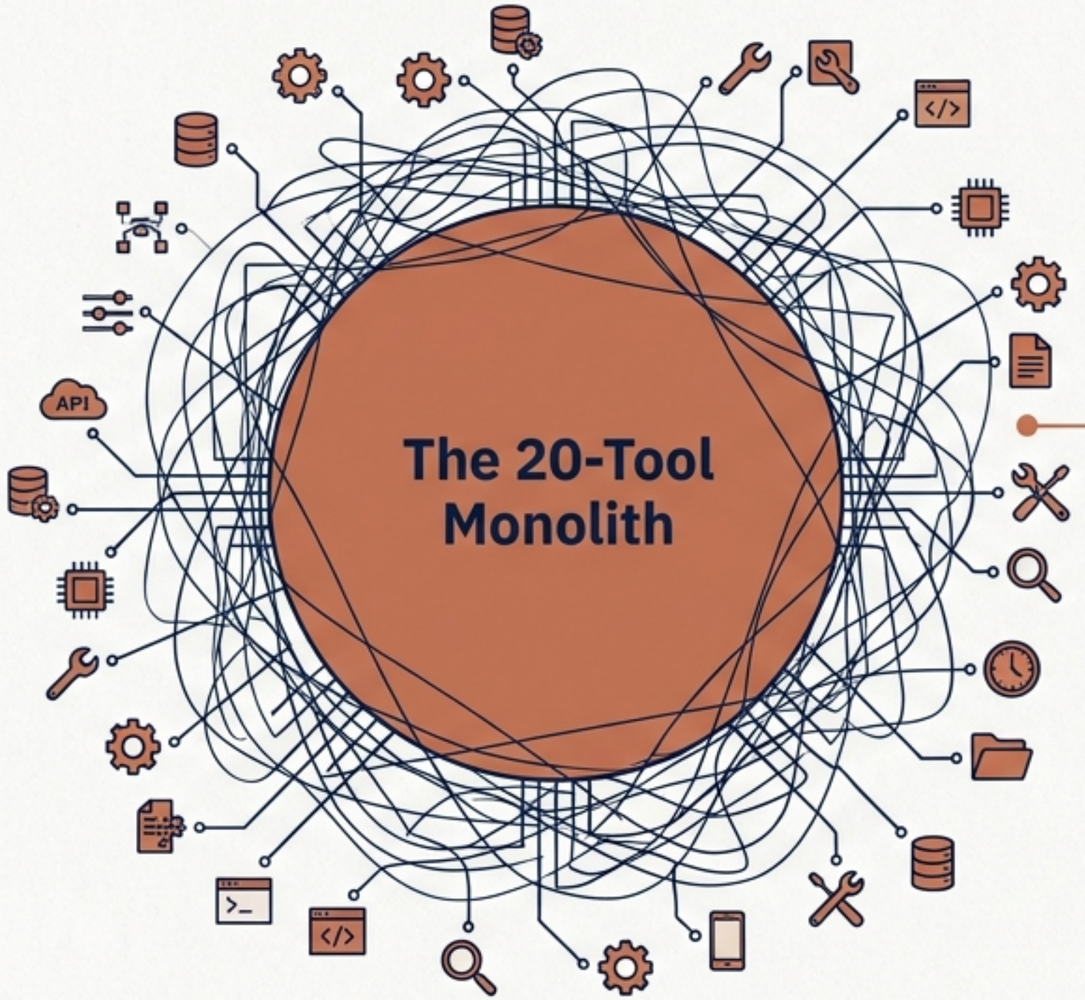


Deconstructing the Monolith

Multi-Agent Orchestration with Vertex AI and GEAP

Production workloads break the single-agent model



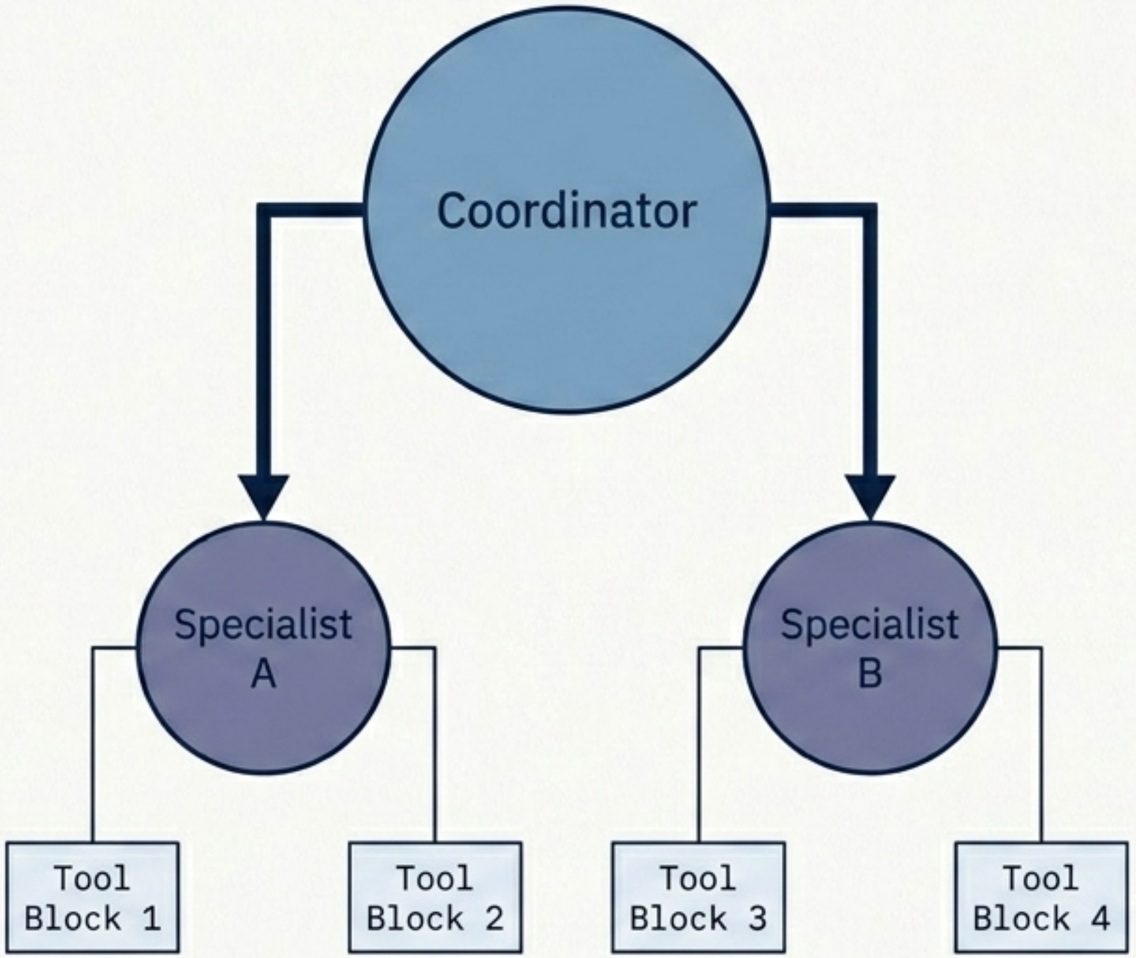
High Risk: Instruction bloat leads to correlated hallucinations.

A single agent accumulates too much instruction length and too many tools, resulting in fragile logic.

The Solution: Decomposition

Split the monolith into specialist agents managed by a central coordinator.

Decomposition creates testable, independently-deployable networks



Platform Native

Since 23 April 2026, Since 23 April 2026, GEAP provides native platform support for agent-to-agent delegation.

Standard Primitives

Sub-agents are standard ADK Agent instances—sharing the same class structure, but with tightly scoped instructions.

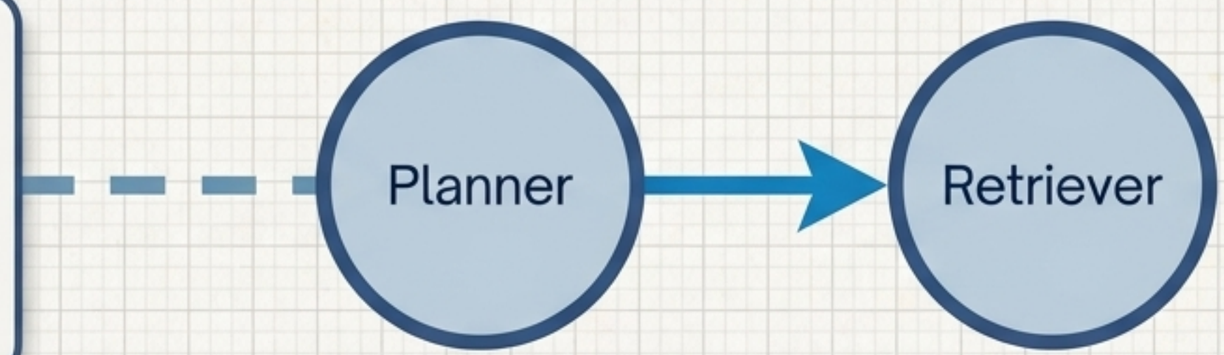
Choose an orchestration pattern based on routing stability

Deterministic Orchestration	Generative Orchestration
`SequentialAgent, ParallelAgent`	`transfer_to_agent(agent_name)`
Hardcoded. Step B always runs after Step A, or both run and merge.	Dynamic. The orchestrator model decides routing at runtime based on user intent.
ETL pipelines, data enrichment, rigid report generation.	Customer support triage, dynamic workflows, intent-based routing.
High (Exact invocation counts are known).	Low (Model decides how many sub-agent calls occur).
Brittle under changing inputs; steps execute even if unnecessary.	Harder to test exhaustively; requires strong orchestrator instructions.

Generative orchestration relies on native transfer primitives

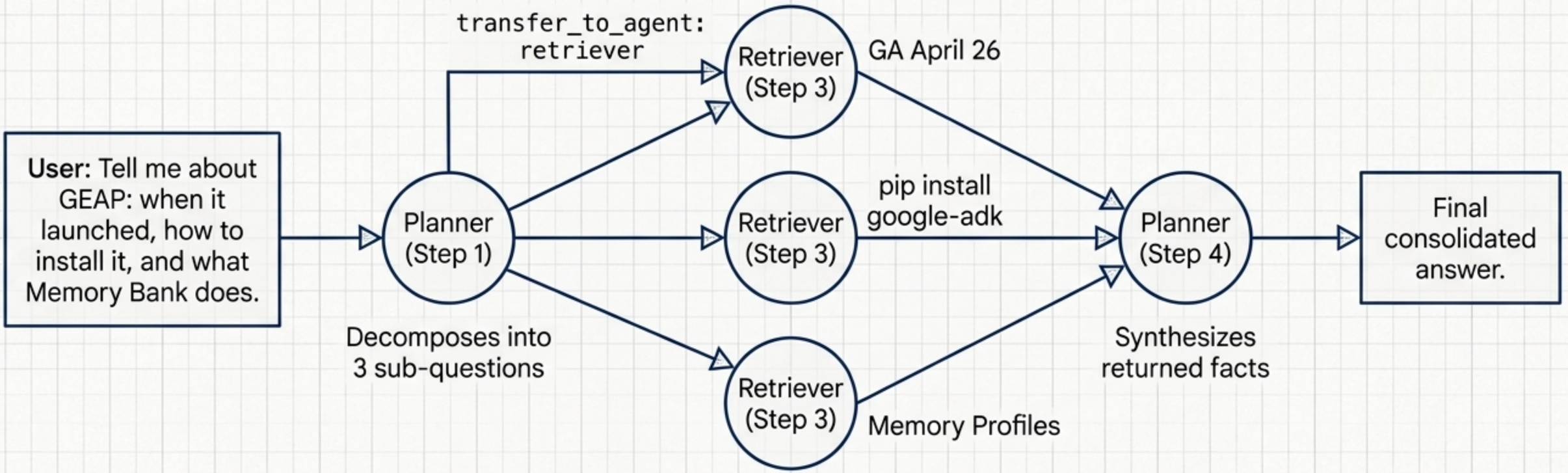
Code Card

```
transfer_to_agent(agent_name="retriever",
message=sub_question)
```



Step 1 block	Step 2 block	Step 3 block
Built-in ADK Tool: Handled by orchestrator like any standard function.	**Execution Pause:** ADK pauses orchestrator and routes payload.	**Context Injection:** Sub-agent returns data directly into reasoning loop.

The generative sequence maps thought directly to execution

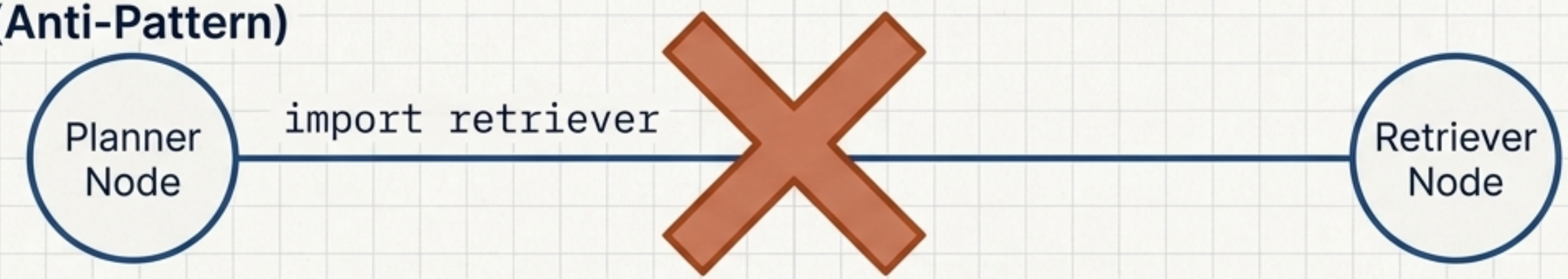


Architecture Note: Memory Bank

Provides long-term cross-session memory by storing distilled Memory Profiles, allowing context recall without raw history bloat.

Production systems decouple discovery from implementation

Local Dev (Anti-Pattern)



Production Vertex (Target Architecture)

Versioning
Update Retriever without touching Planner code.

Governance
Strictly control visibility of approved agents.

Auditability
Create an immutable log of the delegation chain.

Enterprise multi-agent systems require a two-part governance lock

[Agent Registry]

Controls Discovery.
Determines which approved agents are visible by name.

[Agent Gateway]

Controls Execution Security.
Enforces IAM-backed access policies for invocation.

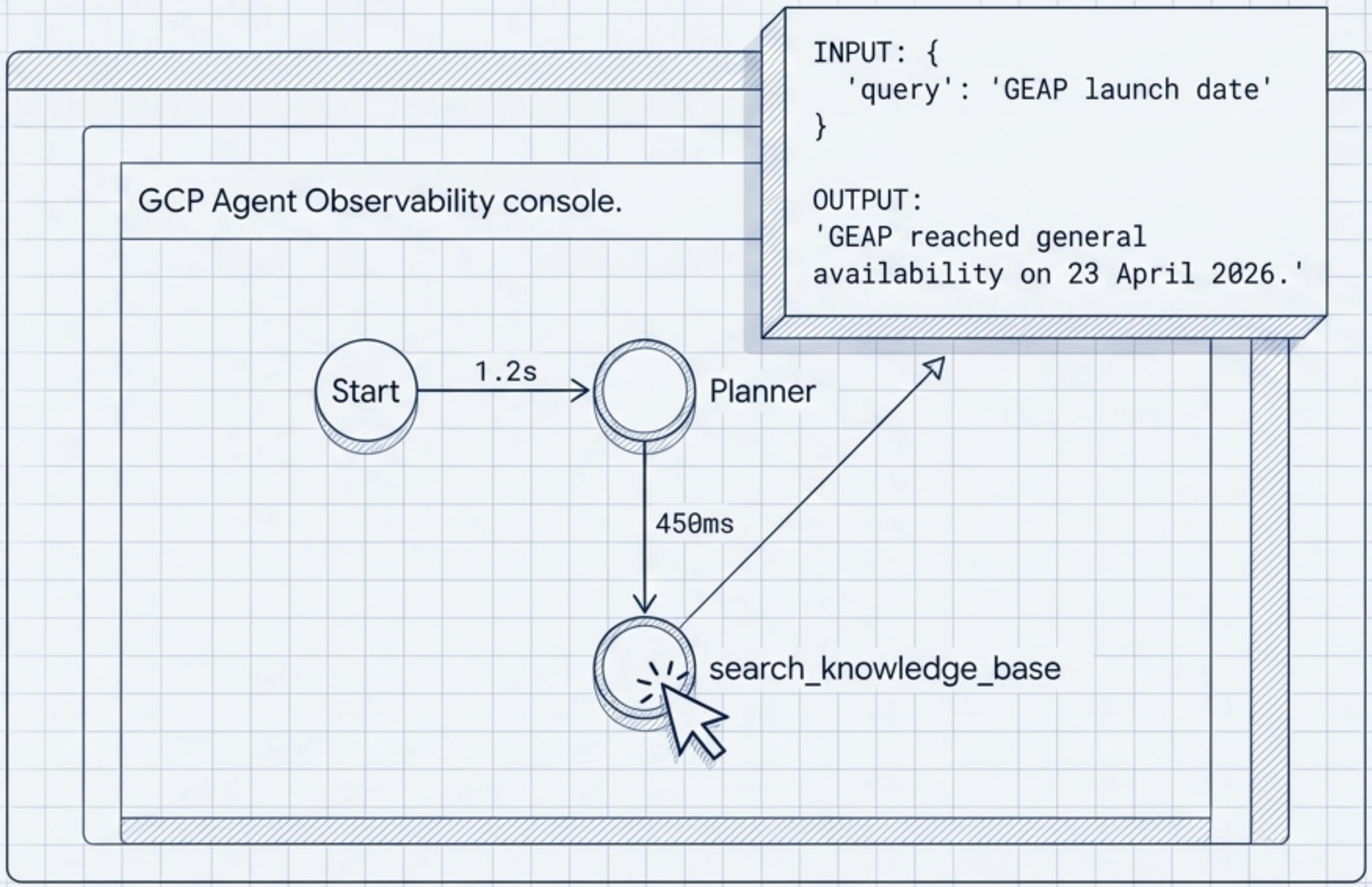
Agent Observability visualizes the entire execution graph

Real-Time Telemetry

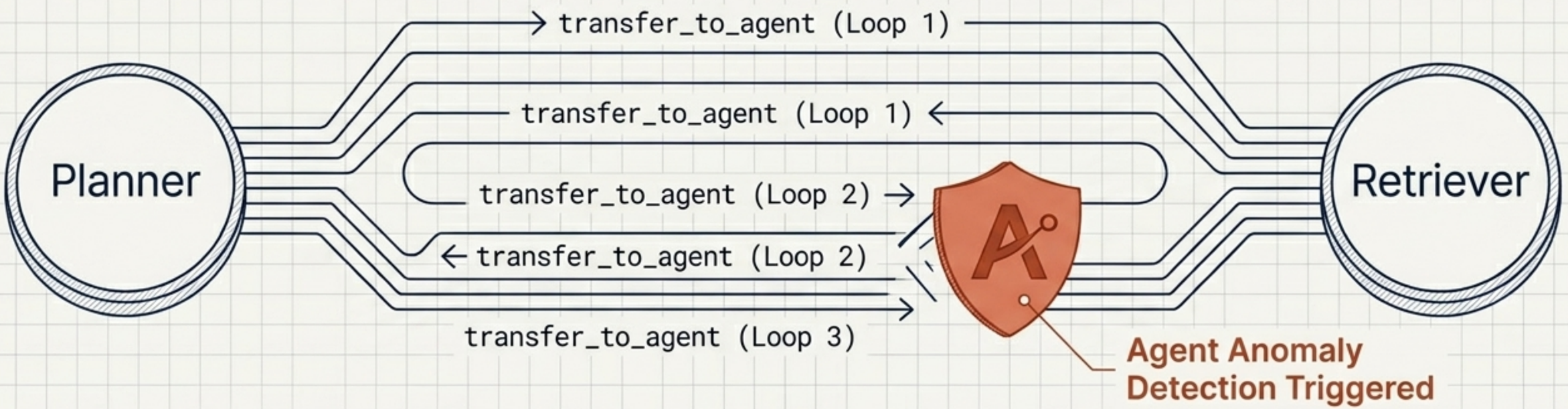
Traces are available within seconds of invocation.

Inspect Every Handoff

Every model call, tool call, and handoff is represented as a clickable node, allowing engineers to inspect the exact payload state anywhere in the delegation chain.



Native anomaly detection severs infinite handoff loops



The Error

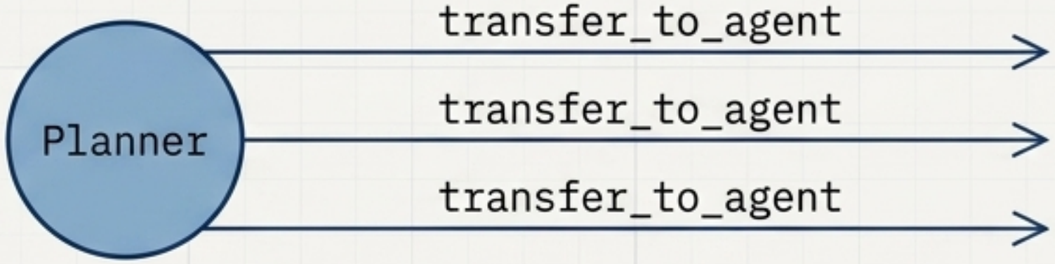
Weak instructions cause the Retriever to inappropriately delegate back to the Planner instead of attempting an answer.

The Guardrail

GEAP automatically monitors reasoning graphs and flags infinite loops within 2-3 hops, requiring zero custom watchdog code.

Silent failures and eager delegations disrupt the sequence

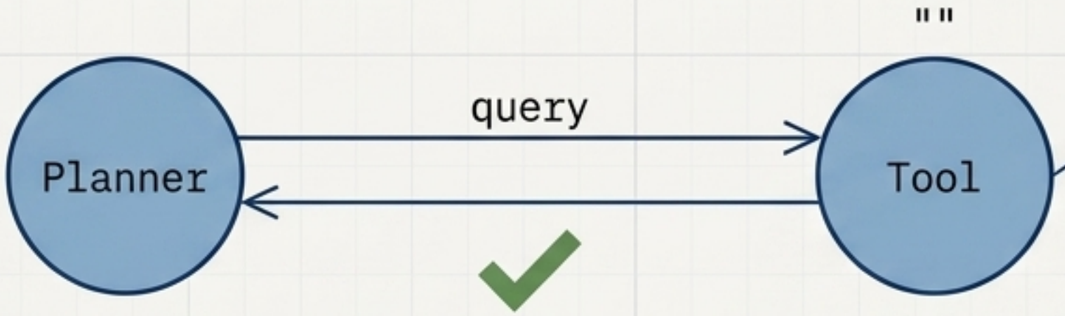
The Premature Fire



Diagnosis:
Missing instruction constraint.

Fix:
Add "wait for the result before the next transfer" to the Planner instructions.

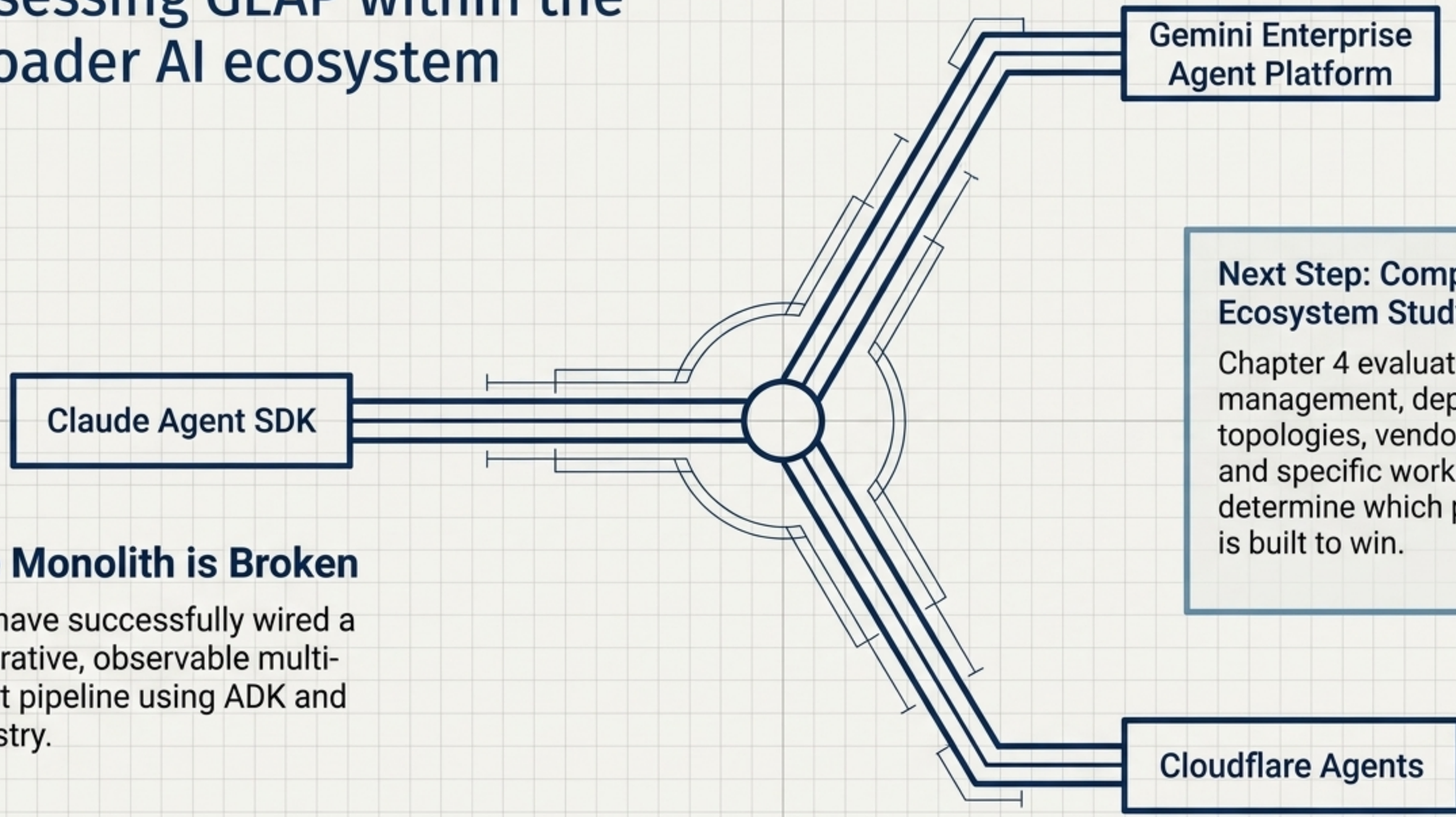
The Silent Failure & Transformation



Diagnosis:
Tools failing silently instead of raising explicit exceptions.

Fix:
Always return explicit “no results” error messages. Tighten Planner to pass queries verbatim.

Assessing GEAP within the broader AI ecosystem



The Monolith is Broken

You have successfully wired a generative, observable multi-agent pipeline using ADK and Registry.

Next Step: Comparative Ecosystem Study

Chapter 4 evaluates state management, deployment topologies, vendor lock-in, and specific workloads to determine which platform is built to win.