

./explain

Agent Design Patterns

Choosing the Right Framework for your Autonomous Workflows

November 2025

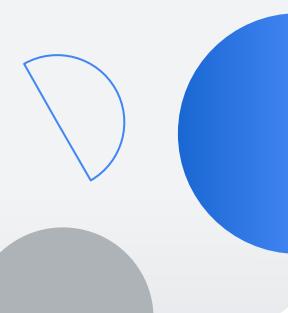


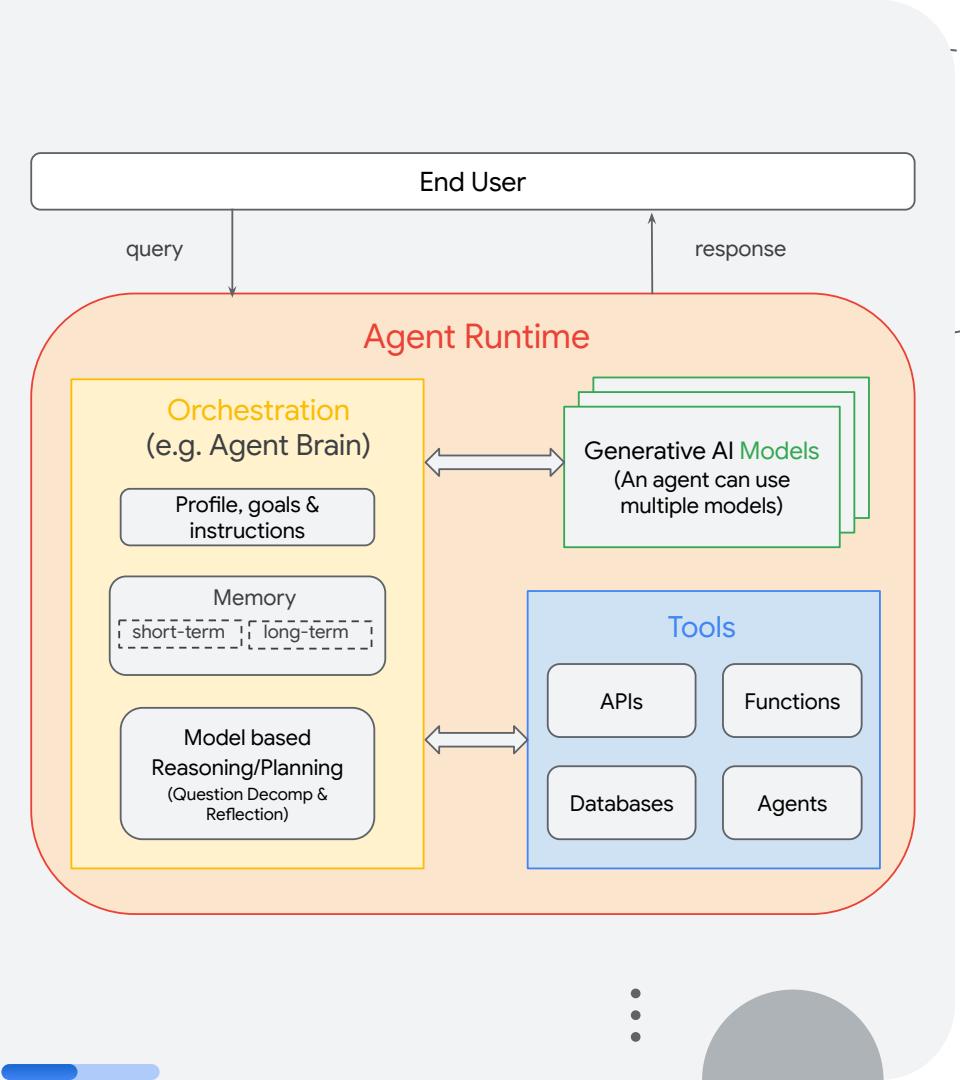
Sanchit Alekh
Customer Engineer, AI/ML

“

An AI Agent is an **application**
that tries to **achieve a goal** by
observing the world and
acting upon it using the tools
it has at its disposal.

:





Building Enterprise-Grade Agents require a robust framework with high software-engineering standards.

Before you start building agents, consider:

1. Code Maintenance
2. Continuous Deployment / Integration
3. Model Variability
4. Answer Quality (Evaluation)
5. AI Safety and Security
6. Monitoring and Traceability

When **NOT** to build Agents

Prompts are Enough

I've got a large prompt that's working well to generate marketing copy. I think I can get by with just **LLM calls**.



Need for highly Exact and Explainable Answers

We run a critical financial transaction system that's heavily regulated. We're going to stick with our **ML classifier** for now.



Q&A on a single source-of-truth

I'm doing Q&A against my enterprise data. I think a **RAG architecture** is enough!



When are Agents a Good Choice?

Need for Autonomy



Complex workflows need autonomy. A human might not be able to comprehend the data available to make a decision and devise an action plan.

Decision-Making Based on Multiple Data Sources



You want **quick insights** from multiple data sources within your organization, but haven't ever gotten to terms with the **data integration complexity**.

High Unpredictability



Traditional RPA systems are **insufficient** for data-based automation and highly unpredictable flows.



Key Considerations for Building Agents



Decision Points

- **Define your requirements:**

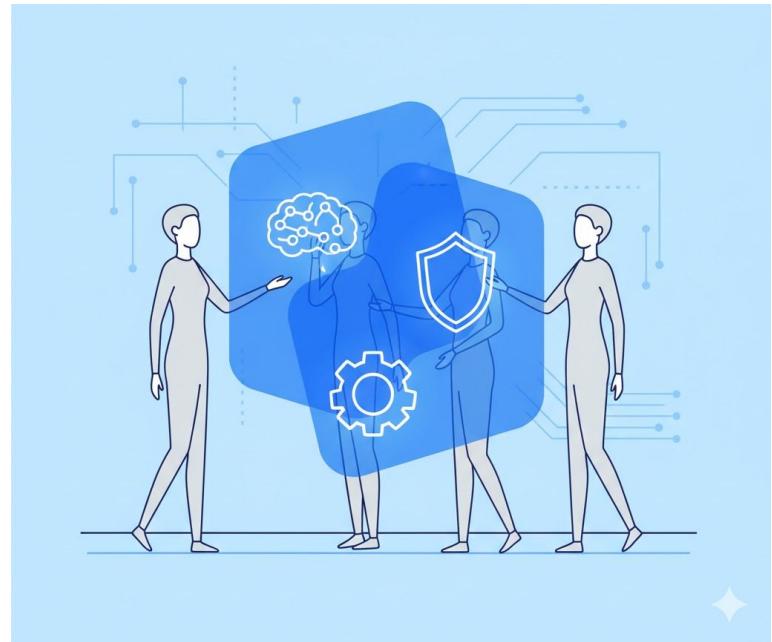
Assess the characteristics of your workload, including task complexity, latency and performance expectations, cost budget, and the need for human involvement.

- **Review the common agent design patterns:**

Learn about the common design patterns in this guide, which include both single-agent systems and multi-agent systems.

- **Select a pattern:**

Select the appropriate design pattern based on your workload characteristics.



Design Patterns

1

Single Agent

Useful for structured, multi-step tasks that require the use of external tools



2

Multi-Agents: Sequential & Parallel

Useful for multi-step tasks that follow a predefined, rigid workflow with dependencies, or to execute tasks in parallel to save time.



3

Multi-Agents: Loop Pattern

Ideal for tasks requiring iterative refinement, like a writer-critic duo for content generation.



4

Multi-Agents: Specific Patterns

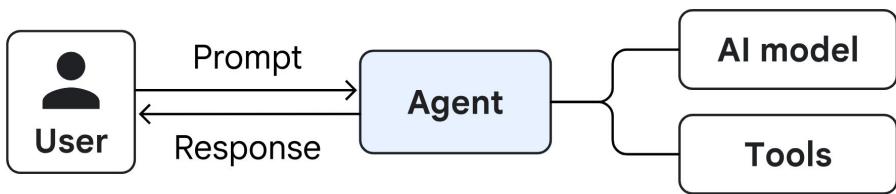
Based on the nature and complexity of the task, other patterns such as Hierarchical Decomposition or Swarm can be used.



Single Agent Pattern

Useful for structured, multi-step tasks that require the use of external tools

- **What:**
One AI model, one prompt, a set of tools.
- **Use For:**
Structured, multi-step tasks with external tools.
The best starting point for agent development.
- **Pros:** Simple, cost-effective, rapid prototyping.
- **Cons:** Limited scalability; performance degrades with too many tools or high complexity.

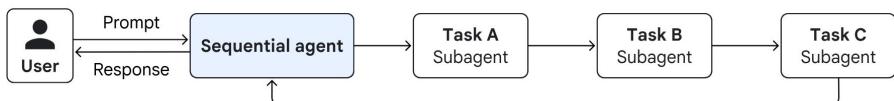


Multi-Agent: Sequential and Parallel

Useful for multi-step tasks that follow a predefined, rigid workflow with dependencies, or to execute tasks in parallel to save time.

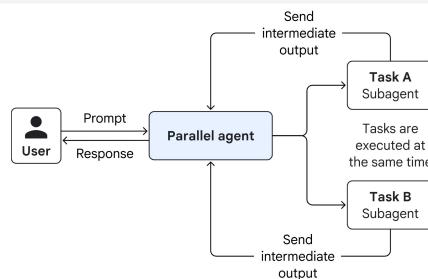
Sequential Pattern

- **What:** Agents in a fixed, linear pipeline. Output of one is input for the next.
- **Use For:**
Rigid, repeatable processes (e.g., ETL pipelines)
- **Pros:** Efficient, predictable, no model needed for orchestration.
- **Cons:** Inflexible; can't adapt or skip steps.



Parallel Pattern

- **What:** Multiple agents that run at the same time. Outputs are combined.
- **Use For:**
Independent sub-tasks to reduce latency or gather diverse views.
- **Pros:** Reduces overall latency.
- **Cons:** Higher immediate cost; results can be complex to merge.



Multi-Agent: Loop

Ideal for tasks requiring iterative refinement, like a writer-critic duo for content generation.

- **What:**

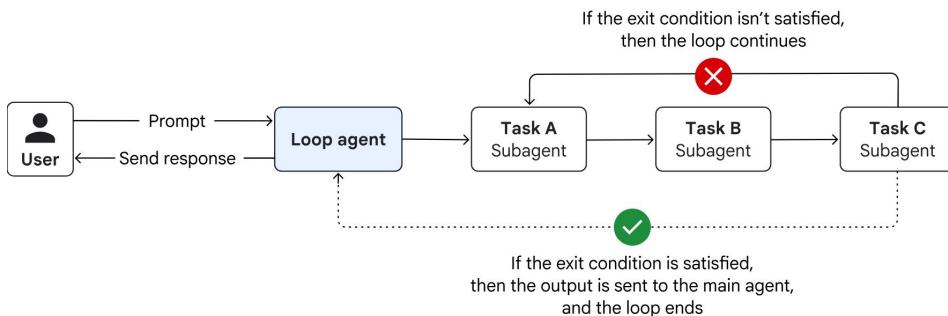
Agents execute repeatedly until a condition is met.
A common variant is Review & Critique (e.g.,
Writer/Critic, or Maker/Checker).

- **Use For:**

Tasks needing iterative refinement and
self-correction (e.g., writing, coding).

- **Pros:** Simple, cost-effective, rapid prototyping.

- **Cons:** High latency and cost; risk of infinite loops.



Multi-Agent: Coordinator / Dispatcher

- **What:**

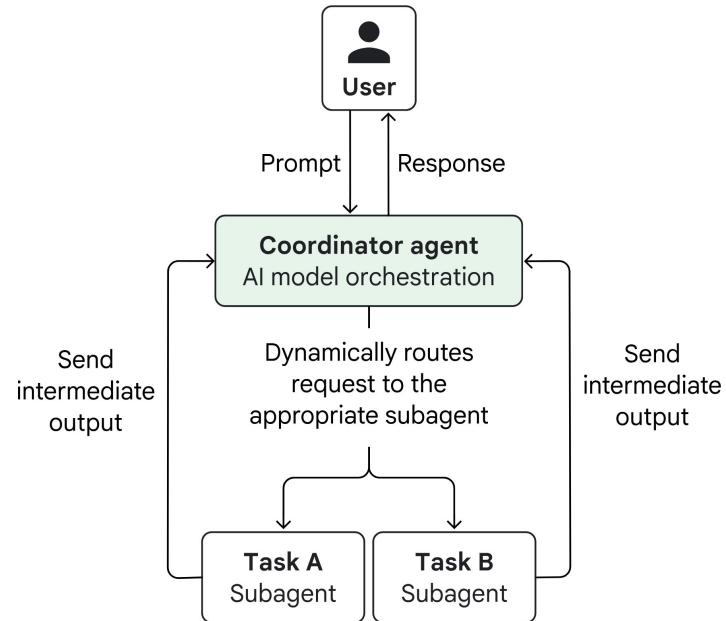
A central agent dynamically routes tasks to specialized agents.

- **Use For:**

Automating structured business processes that need adaptive routing (e.g., a help desk, travel concierge service).

- **Pros:** Flexible and adaptive workflow.

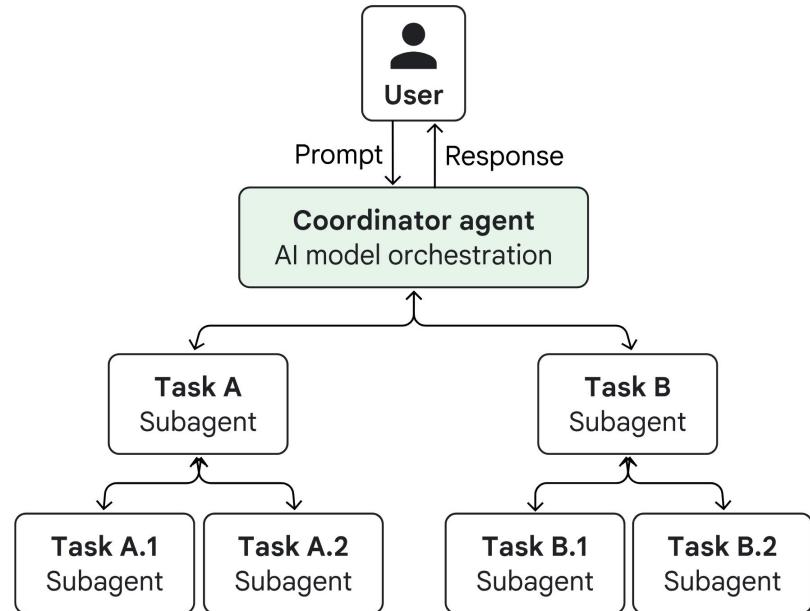
- **Cons:** More model calls mean higher latency and cost vs. a single agent. Use with caution and only when necessary



Multi-Agent: Hierarchical Decomposition

A variant of the coordinator/dispatcher pattern, where the subagents can in-turn become coordinators

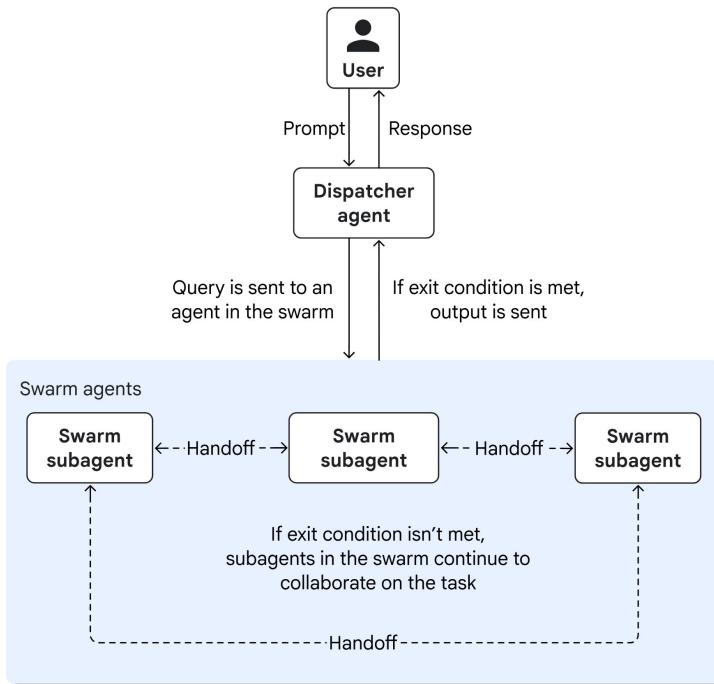
- **What:**
A root agent breaks large, ambiguous tasks into smaller sub-tasks, delegating them down a multi-level hierarchy.
- **Use For:**
Complex, open-ended problems needing significant research, planning, and synthesis.
- **Pros:** Can produce exceptionally creative, high-quality solutions.
- **Cons:** The most complex and expensive pattern. Risk of non-convergence or unproductive loops.



Multi-Agent: Swarm Pattern

A variant of the coordinator/dispatcher pattern, where the dispatcher's role is hands-off after selecting an agent swarm

- **What:**
Multiple specialized agents collaborate with all-to-all communication. No central supervisor.
- **Use For:**
Highly complex or ambiguous problems that benefit from debate and diverse "expert" perspectives.
- **Pros:** Can produce exceptionally creative, high-quality solutions.
- **Cons:** The most complex and expensive pattern. Risk of non-convergence or unproductive loops.



In a Nutshell

Simple and Structured Tasks

Start with a **Single Agent**

Predictable or Rigid Workflow

Use **Sequential** or **Parallel** Workflows

Needs Iterative Improvement or Quality Control

Use the **Loop** Pattern or **Coordinator / Dispatcher** Pattern

Massive and Ambiguous Problem

Hierarchical Decomposition (for deep planning)
Swarm (for creative, debate-driven solutions)



Thank you!

For detailed architectural guidance for building Agentic solutions on Google Cloud, refer to [Cloud Architecture Center](#)

