What are Multi-Agent Coordination patterns?

Multi-Agent Coordination Patterns are architectural and interaction models that define how multiple independent and specialized AI agents (such as Large Language Model-based agents) organize, communicate, and collaborate to achieve a complex, shared objective.

These patterns are essential for building **Multi-Agent Systems (MAS)**, which break down intricate problems into subtasks that can be solved more efficiently by a team of specialized agents rather than a single general-purpose agent.

Commonly used multi-agent coordination patterns include:

- **Hierarchical Task Decomposition (Supervisor Pattern):**
    - **How it works:** A high-level **supervisor** or **lead agent** is responsible for overall workflow and task delegation. It breaks the complex task into smaller subtasks and delegates them to specialized **worker agents**. The supervisor then integrates the results.
    - **Control Flow:** Centralized, with a clear chain of command.
    - **Example:** A Research Agent that plans the overall research, creates sub-agents to search for different aspects in parallel, and then synthesizes the final report.
- **Sequential Pipeline Pattern (Deterministic Flow):**
    - **How it works:** Agents are arranged in a specific, fixed sequence. The output of one agent becomes the input for the next, like an assembly line.
    - **Control Flow:** Sequential, synchronous transfer of data and control.
    - **Example:** A workflow where a **Triage Agent** classifies a customer request, which is passed to a **Technical Researcher Agent**, which then hands the context to a **Response Crafter Agent** for the final output.
- **Parallel Fan-Out/Gather Pattern (Router & Parallel):**
    - **How it works:** A central agent dispatches independent subtasks to multiple agents to execute simultaneously (fan-out). Once all parallel tasks are complete, the central agent collects (gathers) and aggregates the results.
    - **Control Flow:** Centralized dispatch, parallel execution, centralized finalization.
    - **Example:** Searching for a topic where an agent sends queries to a Web Search Agent, a Database Agent, and a Code Executor Agent all at the same time to gather comprehensive information.
- **Handoff/Agent Transfer:**
    - **How it works:** The current active agent decides that another agent is better equipped to handle the next part of the task or conversation and explicitly transfers control and context to that agent.
    - **Control Flow:** Decentralized and dynamic, changing the active agent interacting with the user or managing the task.

    - **Example:** A general customer service agent transferring a user to a specialized **Billing Agent** when the user's query shifts to account payments.
- **Review/Critique Pattern (Generator-Critic):**
    - **How it works:** One agent (**Generator**) creates an output (e.g., a piece of code or a design plan), and a second agent (**Critic** or **Reviewer**) critiques it, looking for errors, improvements, or security flaws. The process often iterates until the Critic approves the output.
    - **Control Flow:** Iterative loop between two specialized agents.
    - **Example:** A **Code Generation Agent** writes a function, and a **Security Auditing Agent** reviews the code for vulnerabilities, sending it back to the generator with feedback.

These patterns provide the structure needed to manage state, communication, and decision-making in systems where autonomy and coordination are crucial for tackling real-world, dynamic problems.

Azure AI Foundry

The **Azure AI Foundry Agent Service** is a **fully managed cloud platform** by Microsoft Azure for building and deploying production-ready, intelligent AI agents.

It simplifies the process by handling all the complex engineering required for an agent to work, essentially providing a serverless runtime environment for agent workflows.

Key Functions

1. **Orchestration:** It manages how multiple agents coordinate and execute complex, multi-step tasks (like the sequential or hierarchical patterns we discussed earlier).
2. **Tooling & Actions:** It enables agents to securely use external tools (APIs, Azure Functions, custom code) and knowledge bases (RAG via Azure AI Search) to take actions and stay grounded in current, proprietary data.
3. **Enterprise Grade:** It provides the necessary security, logging, governance, and scalability required for large organizations.

Simply Put

It takes the concept of an AI agent—a piece of software that can reason, plan, and act autonomously—and turns it into a **reliable, scalable cloud resource** that developers can easily integrate into their applications, without having to manage the underlying infrastructure.