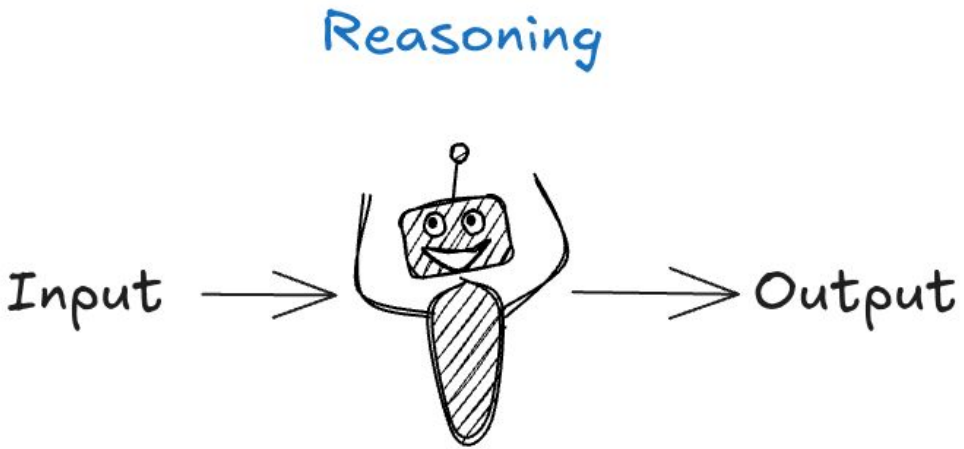
AI Agent Architectures

Md. Shahad Mahmud Chowdhury
AI Team Lead
Delineate (YC W25)

AI Agent

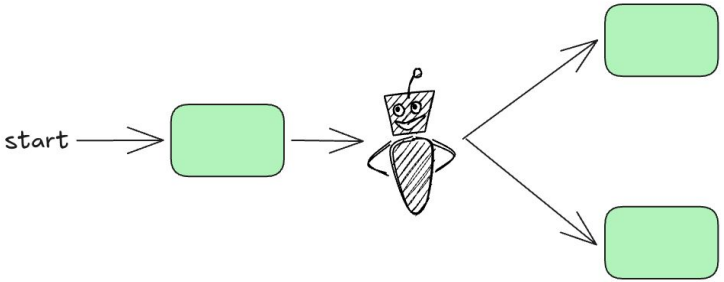
An LLM powered entity that has capability of reasoning, perception (with inputs) and action (with tools). It uses an LLM to decide the control flow of an application.



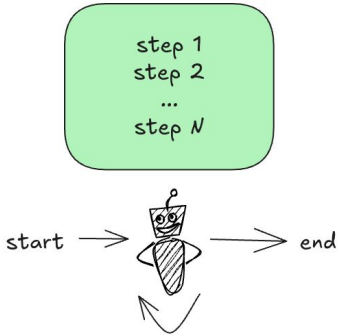
Agent Architecture

Agent architecture (or agentic architecture) refers to the structure and design of simple to complex agentic systems outlining the components and their interactions.

The architectural components particularly outlines the level of control or autonomy of an AI agent.



Agent can decide to take a step between two defined options



Agent can decide to take any step on its own.

Bottom Line Up Front

- Core Components
  - Memory
  - Tool Calling
  - Planning
- Decision Making
- Multi-Agent Architectures
- How Multi-Agents Collaborates
- Message Passing Between Agents
- Handoffs

Memory

Memory is a system that remembers things about *previous interactions*.

Unlike traditional AI models that process each task independently, AI agents with memory can retain context, recognize patterns over time and adapt based on past interactions.

We generally have two types of memories:

1. Short Term Memory (STM)
2. Long Term Memory (LTM)

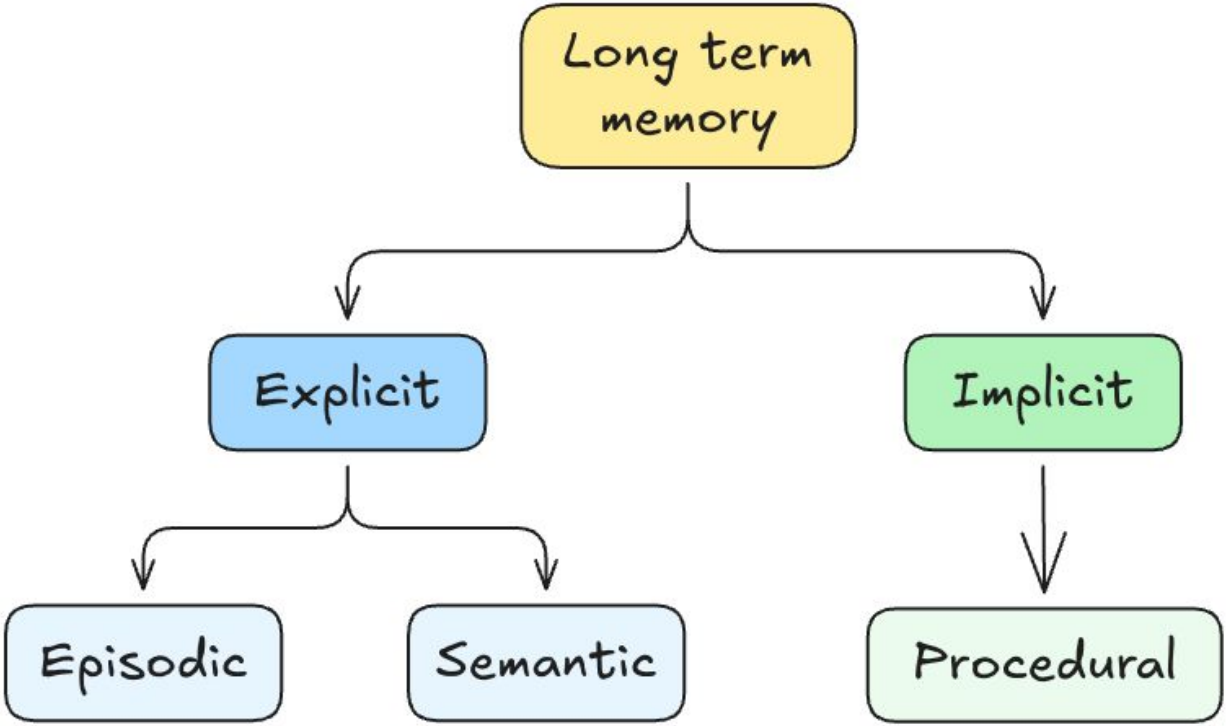
Short Term Memory

- Short-term memory (STM) in AI:
  - Enables agents to recall recent inputs for immediate decisions.
  - Crucial for conversational AI to maintain context across exchanges.
- Implementation and limitations:
  - Commonly uses a rolling buffer or context window for recent data.
  - Enhances continuity in short interactions but doesn't retain information beyond the session.
  - Unsuitable for long-term personalization or learning.

Long Term Memory

- Long-term memory (LTM) enables AI agents to store and recall information across different sessions.
  - Fosters personalization and intelligence.
  - Unlike short-term memory, LTM is for permanent storage.
    - Often uses databases, knowledge graphs, or vector embeddings.
- Crucial for AI applications needing historical knowledge.
  - Examples: personalized assistants, recommendation systems.
- Retrieval augmented generation (RAG) is an effective LTM implementation technique.
  - Agents fetch relevant information from a knowledge base to enhance responses.

Long Term Memory: Memory Types



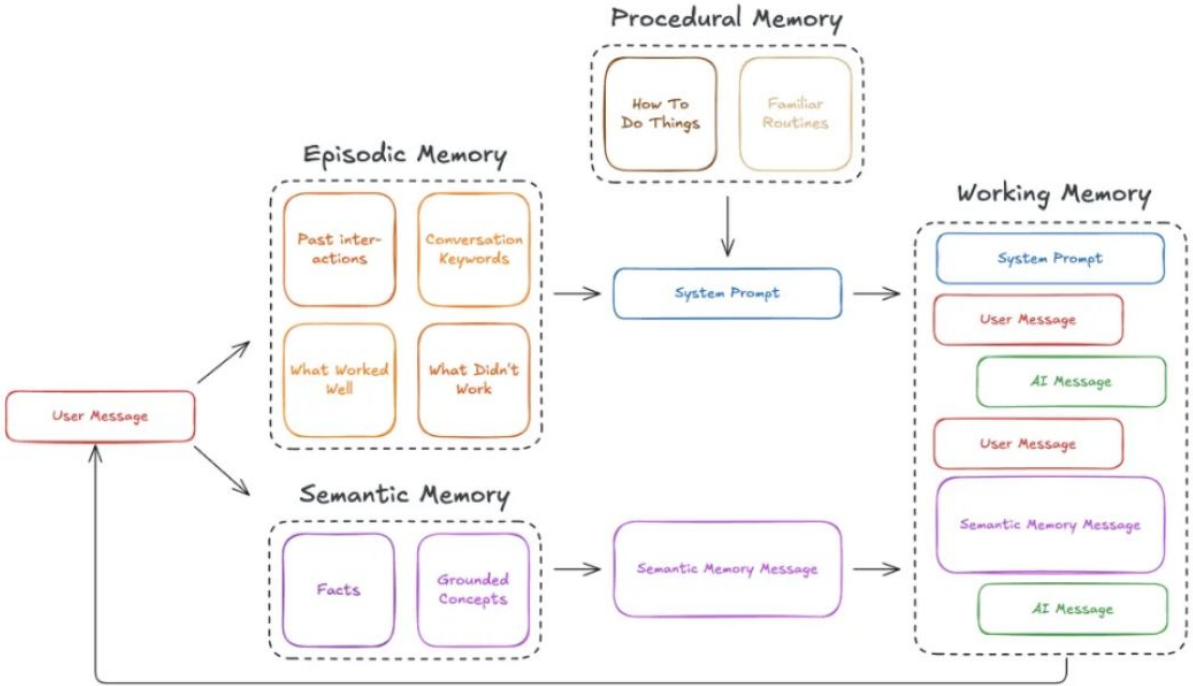
Long Term Memory: Memory Types

- **Episodic memory:** It enables AI agents to recall specific past experiences, like human event recall.
  - Useful for case-based reasoning, AI learns from past events for future decision-making.
  - Implemented by logging key events, actions, and outcomes in a structured format for agent access.
  - For instance, an AI financial advisor uses past investment choices for better recommendations.
- **Semantic memory** stores structured factual knowledge for AI reasoning.
  - Unlike episodic memory, it contains generalized information like facts, definitions, and rules.
  - AI agents implement it via knowledge bases, vector embeddings, etc. for efficient information retrieval
  - Used in applications like legal AI, medical diagnostics, and enterprise knowledge management.

Long Term Memory: Memory Types

- **Procedural memory** in AI allows agents to store and recall learned skills and rules, enabling automatic task performance without explicit reasoning.
  - Inspired by human procedural memory (e.g., riding a bike), it boosts efficiency by automating action sequences based on experience.
  - AI agents learn these sequences through training, often with reinforcement learning
  - Reduces computation and speeding up responses by not reprocessing data from scratch.

Memory: Bringing all together



Full Working Memory Demonstration

Taken from [HuggingFace]

Tool Calling

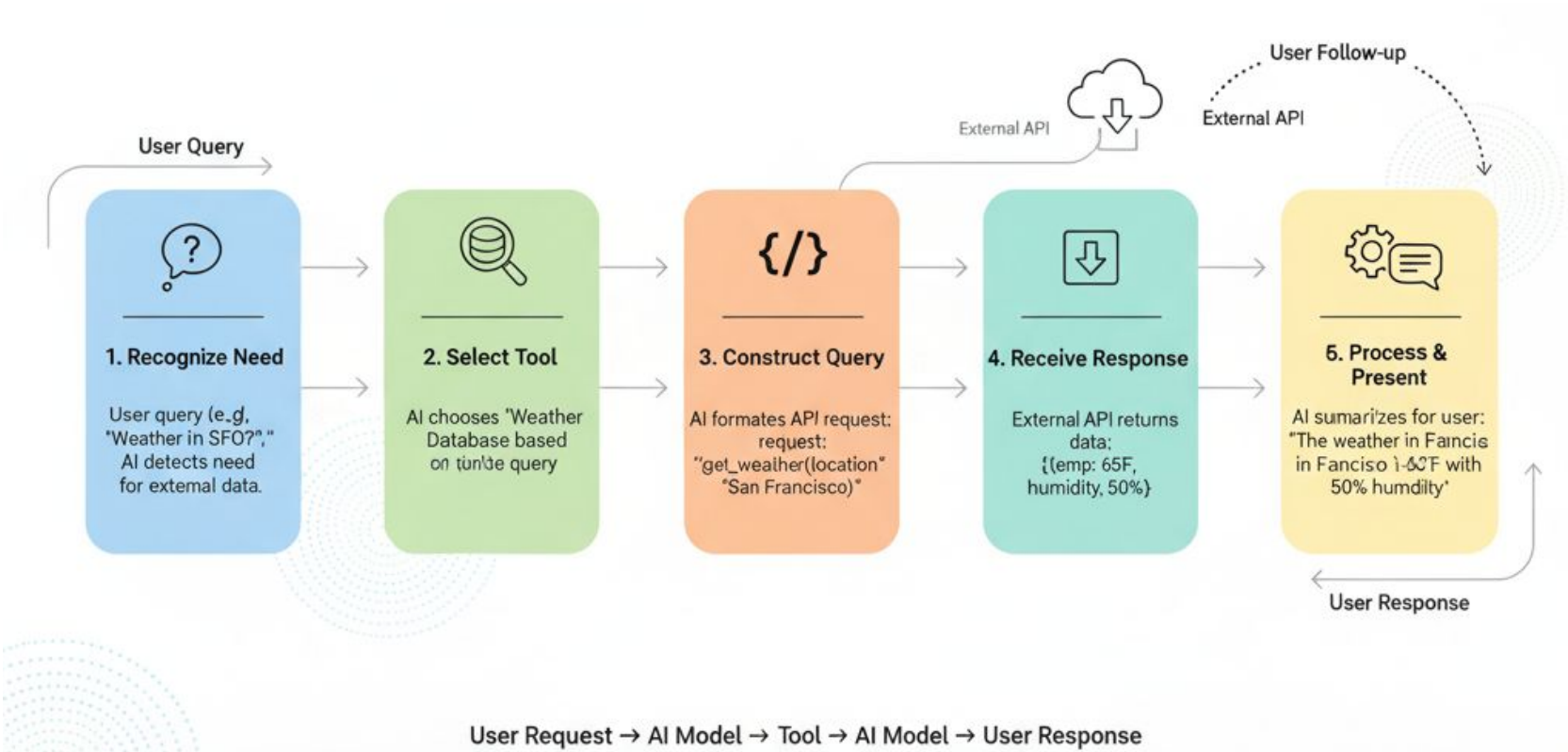
Tool calling (or function calling) enables AI models to interact with external tools, APIs, or systems, expanding their functionalities.

Rather than relying solely on pre-trained knowledge, an AI system with tool-calling capabilities can **query** databases, **retrieve** real-time information, **execute** functions, or **perform** complex operations beyond its inherent capabilities.

Tool Call: How does it work?

- Tool calling enables AI models like Claude, Llama 3, Mistral, etc. to interact with external tools.
- The AI model identifies when external functions are needed.
  - A tool selection mechanism then identifies the appropriate tool (e.g., search engine, database).
  - The API interface facilitates sending queries and receiving responses.
  - Finally, the response processing system formats the data and presents to the user.
- A tool is mainly a function running some custom logic.

Tool Call: How does it work?

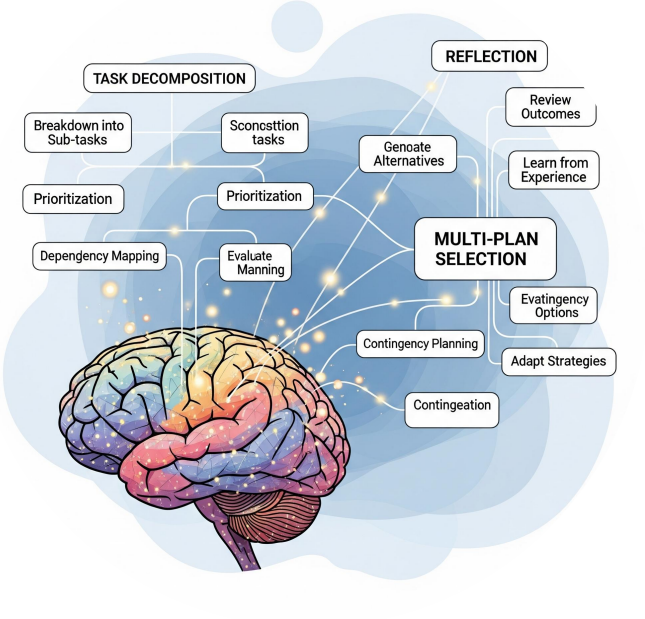


Planning

The planning module serves as the agent's core decision-making unit, defining optimal action sequences for goal attainment.

- **Goal-Oriented Behavior:** Deliberative AI (planning agents) pursue specific goals, using planning algorithms to assess actions.
- **Information Utilization:** The planning module analyzes the current situation, drawing on memory and perception (or profiling) for effective action.
- **Hierarchical Control:** Planning simplifies complex tasks by breaking high-level goals into sub-goals and action sequences, crucial for managing complexity.

Planning: Approaches



- **Task decomposition** simplifies complex goals into subtasks.
- **Multi-plan selection** chooses optimal plan from options.
- **External module aided planning** leverage existing or specialized plans.
- **Reflection** refines plans with new feedback or information.
- **Memory-augmented planning** uses stored information to improve plans.

Decision Making

AI agents' decision-making capabilities fall generally into two main categories: **Reactive** and **Deliberative**.

- Reactive agents operate based on predefined rules or behaviors that directly map perceptions to actions (responses).
- Deliberative agents (also known as cognitive or planning agents) make decisions based on complex reasoning, planning, and maintaining an internal state.

Decision Making: Reactive Agents

They make decisions in real-time without complex reasoning or planning.

Perception-Action Loop: The core principle is the direct coupling of perception and action. When a stimulus is perceived, a corresponding action is immediately triggered.

State: They typically do not maintain an internal state or memory of past actions; their decisions are based solely on the current environmental state.

Use Cases: Reactive agents are ideal for fully observable, stable environments and tasks requiring quick, real-time responses, such as customer service chatbots, simple robotic obstacle avoidance, etc.

Decision Making: Deliberative Agents

Makes decisions based on complex reasoning, planning, and maintaining an internal representation of the world.

Process: They are goal-oriented, reasoning about the best actions to achieve specific goals, considering the future consequences of their actions to optimize performance over time.

Planning and Reasoning: These agents use planning algorithms, often employing search and optimization techniques, to explore possible action sequences and choose the best path.

Internal States: They maintain internal models of their environment, which they use to simulate and predict the outcomes of different actions.

Decision-Making Pattern: The process follows a more complex Perception-Decision-Action Loop: they perceive the environment, update their internal state, deliberate (plan and reason), and then act.

Use Cases: Deliberative agents handle complex tasks requiring strategic thinking and long-term goal achievement, such as autonomous vehicles, financial trading systems, and healthcare treatment planning

Multi-Agent Architectures

As the AI agent systems grow, they can become complex and difficult to manage, leading to issues such as:

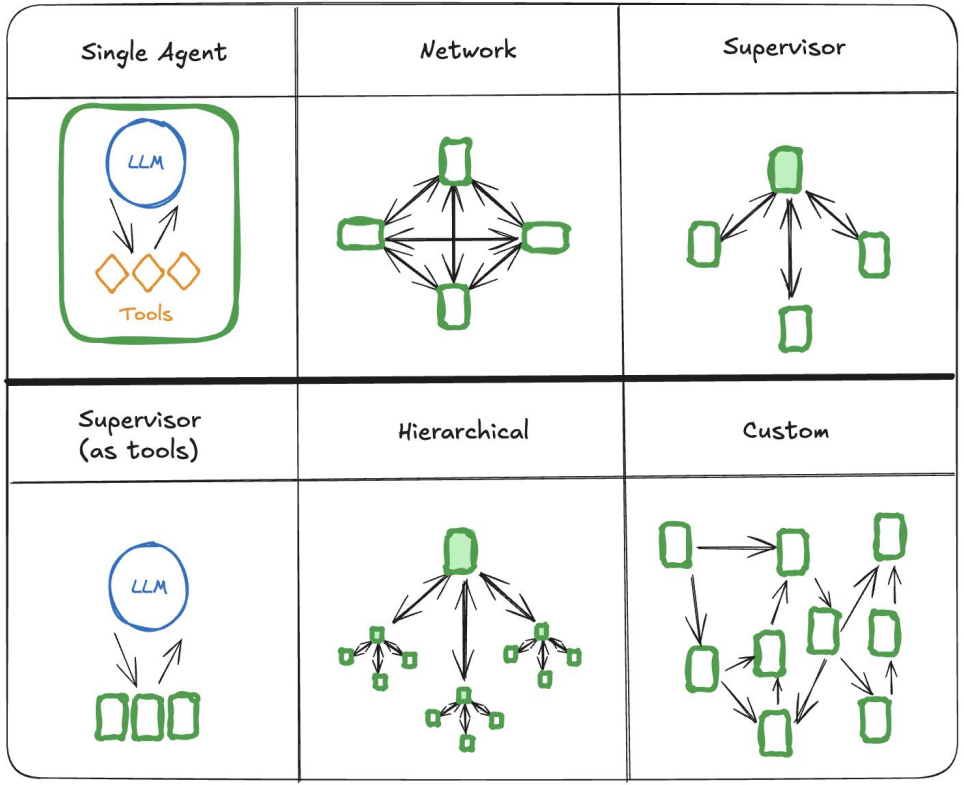
- Too many tools, resulting in poor decision-making.
- Overly complex context for a single agent.
- Need for specialized areas (e.g., planner, researcher, math expert).

To address these, we can break applications into smaller, independent agents and compose them into a multi-agent system.

Benefits of multi-agents systems are:

- Modularity
- Specialization
- Control

Types of Multi-Agent Architectures

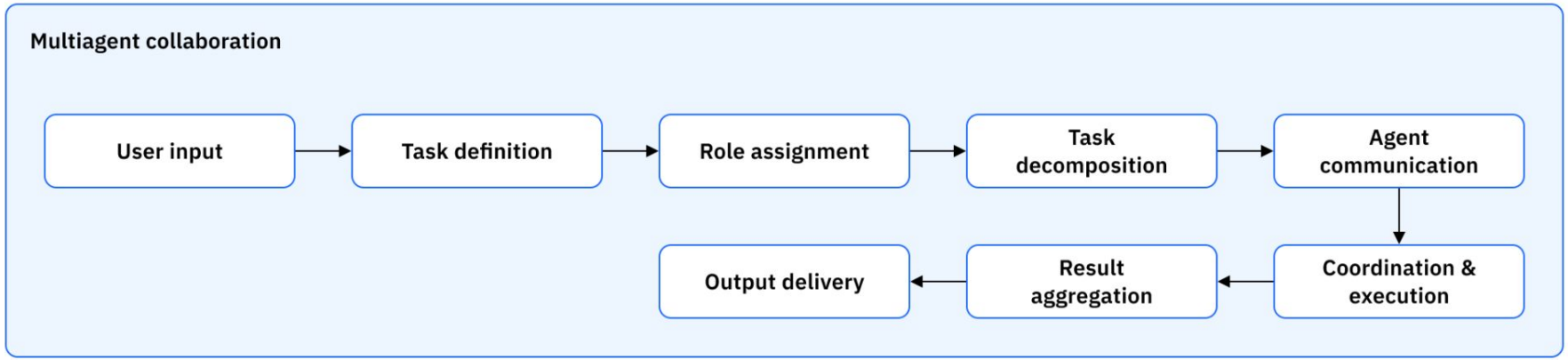


How Multi-Agents Collaborates

Agents collaborate via structured channels, with each intelligent agent having five key elements:

1. **Foundation model:** The main reasoning engine for natural language generation and comprehension.
2. **Objective:** Defines the agent's goal or task.
3. **Environment:** The agent's operational situation, including other agents, tools, shared memory, or APIs.
4. **Input perception:** Information received from surroundings or other agents.
5. **Output or Action:** The agent's response based on its objective and reasoning.

How Multi-Agents Collaborates



AI agents collaborate as a team to accomplish tasks. The system receives a task, determines necessary agents and roles, and divides complex problems into manageable pieces using a planner or LLM with reasoning.

Communication occurs through shared memory or intermediate outputs, with agents performing assigned tasks concurrently, sequentially, or dynamically.

Communication and State Management

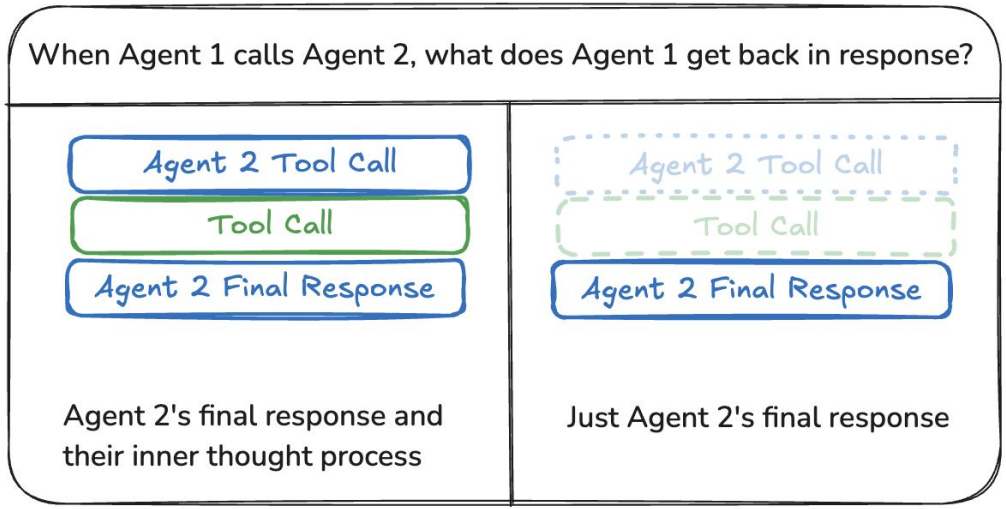
The most important thing when building multi-agent systems is figuring out how the agents communicate.

A common, generic way for agents to communicate is via a list of messages.

Message Passing Between Agents

Agents usually communicate through a shared state channel, like a message list.

This means a single shared channel (e.g., "messages") must exist in the state. When using a shared message list, consider if agents should share their full thought process or just the final result.



Message Passing Between Agents

Sharing full thought process: Agents share complete thought history (list of messages) with all agents. Benefits include improved decision-making and system-wide reasoning. Downside: message lists grow quickly with more agents, requiring memory management strategies.

Sharing only final results: Agents maintain private message lists, sharing only the final outcome. This suits systems with many or complex agents, requiring agents with distinct state schemas.

It can be helpful to indicate which agent a particular AI message is from.

Handoffs

A common pattern in multi-agent interactions is handoffs, where one agent hands off control to another. Handoffs allow us to specify:

- destination: target agent to navigate to like to go agent bob
- payload: information to pass to that agent like passing user query

Tools can perform handoffs too. Like execute a certain task and pass the result to a specific agent.

Thank you!

References

1. Hugging Face memory blog
2. IBM tool calling guide
3. https://langchain-ai.github.io/langgraph/concepts/multi_agent/
4. https://www.ibm.com/think/topics/multi-agent-collaboration#2014952963