/OSEDEA

Building Super Agents with Micro-Agent Orchestration

Innovation and ideas made real

Multi-Agent Systems (MAS)

In this session, we will explore what it takes to setup **micro-agents** and **orchestrating** them to form a versatile **super agent**. This approach leverages the strengths of each micro-agent, which possess their own planning, reflection, and self-critique capabilities, to collectively tackle complex tasks.

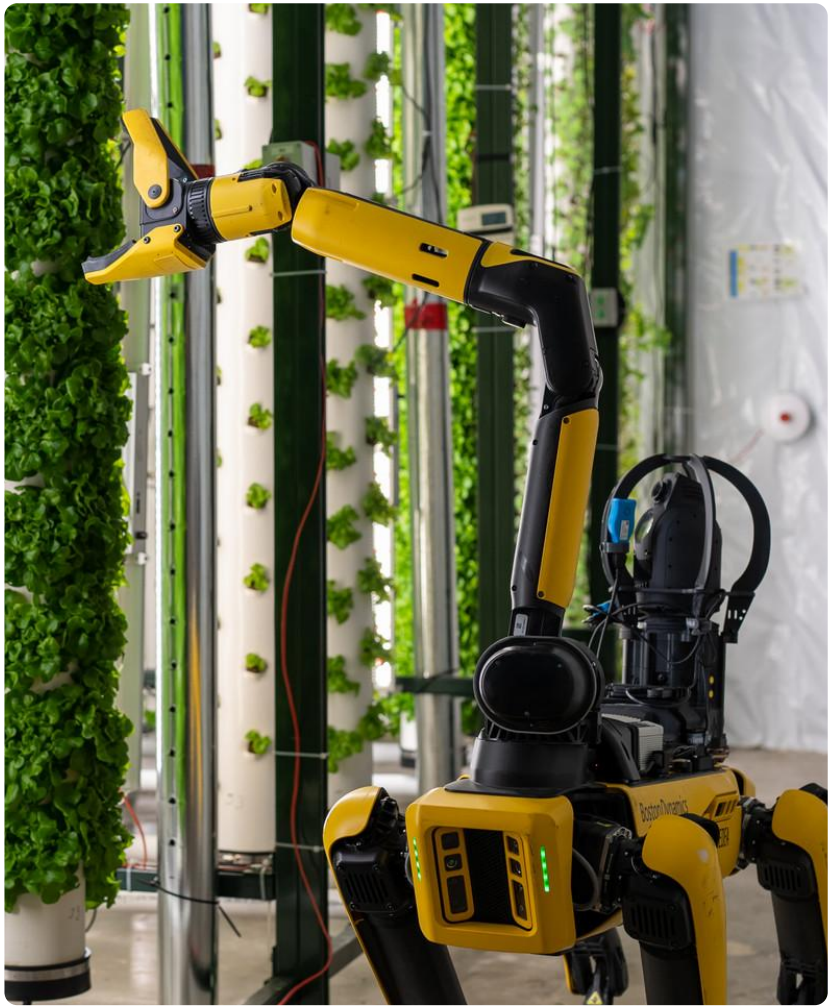
Today's agenda

- Introductions
- Some MAS definitions
- Control Flow Patterns
- Control Flow Communication
- Control Flow Protocols
- Demo

Hi! I'm Carl Lapierre

- Software developer from **Montreal, Quebec, Canada, Earth**
- 10+ years of experience in software
- Rich background in the education and online learning industry
- Developed search solutions
- Created digitization workflows
- Built content management platforms
- And more!

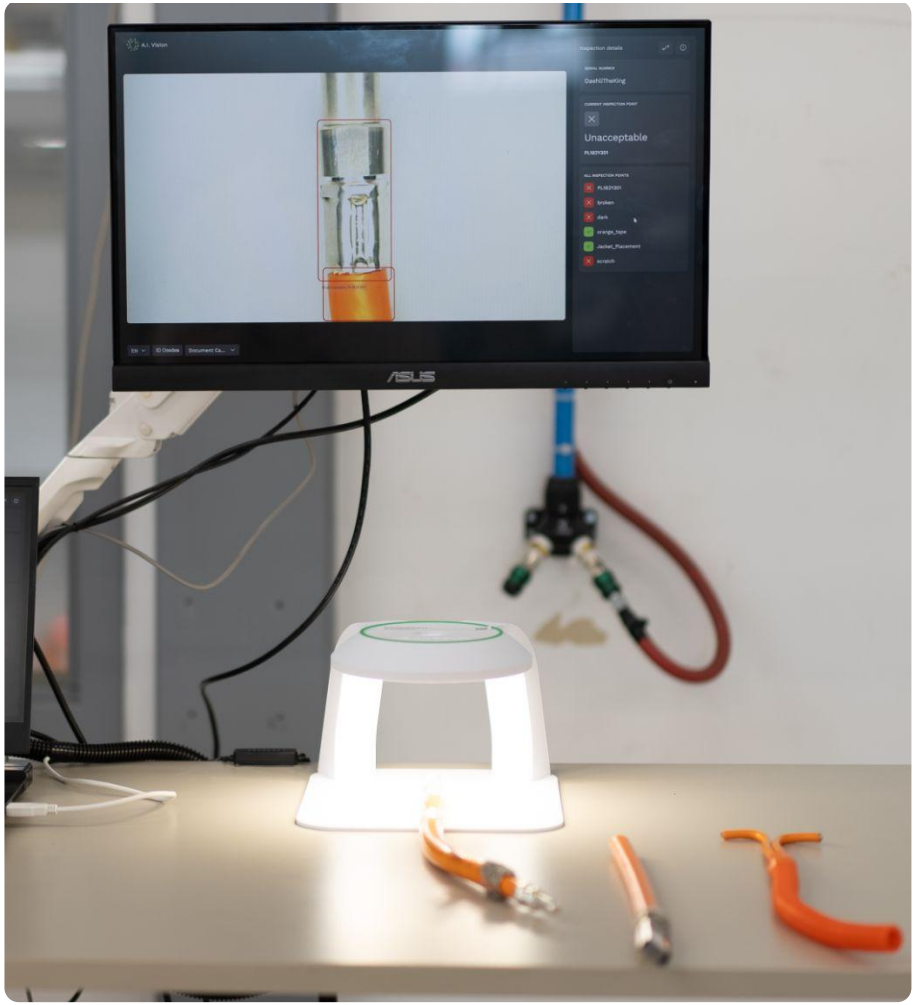
carllapierre

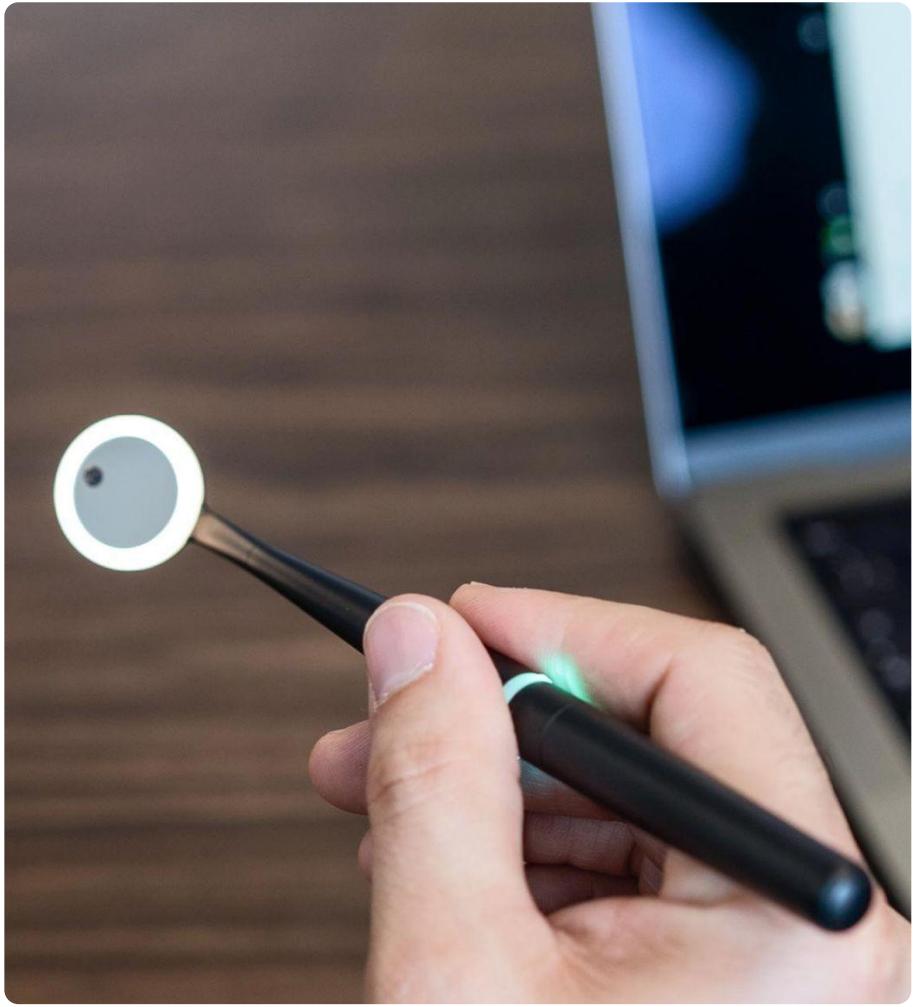


/OSEDEA

Osedea is a highly regarded innovation firm in Montreal that combines creativity, technical expertise, and passion to bring cutting-edge solutions to life.

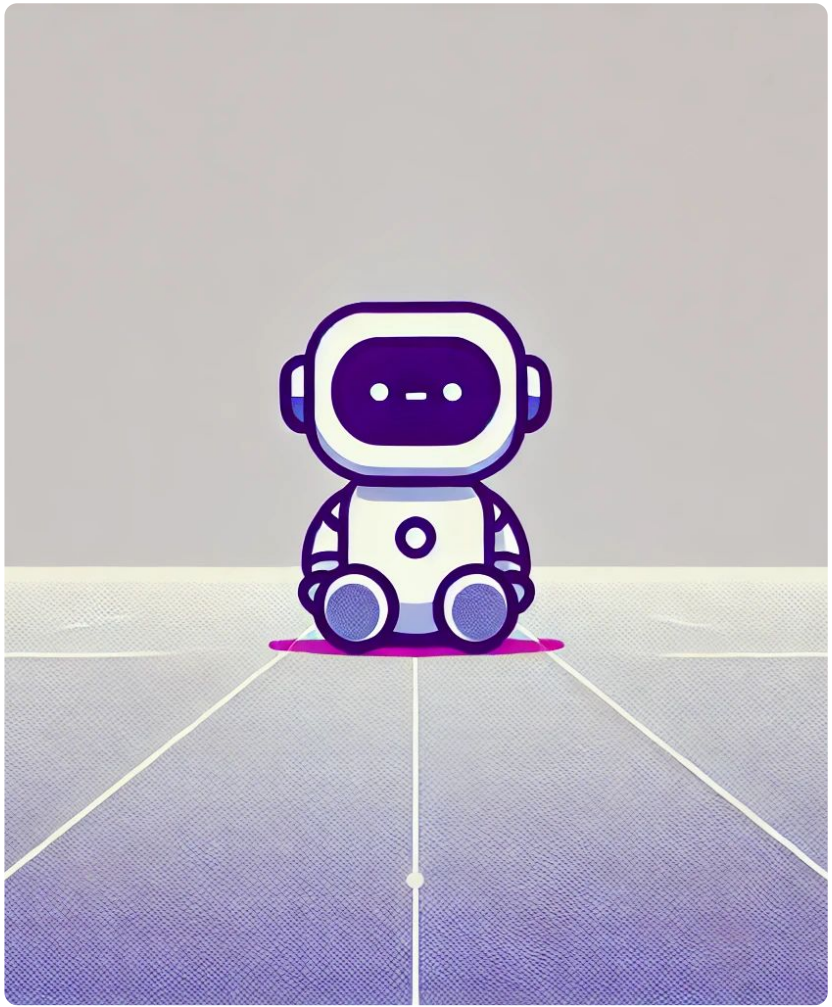
- Product design
- Software development
- Artificial intelligence
- Robotics


A.I. Vision
Unacceptable
broken
dark
orange_tape
Jacket_Placement
scratch
ASUS



Let’s define a few things

Agents and Multi-Agent Systems

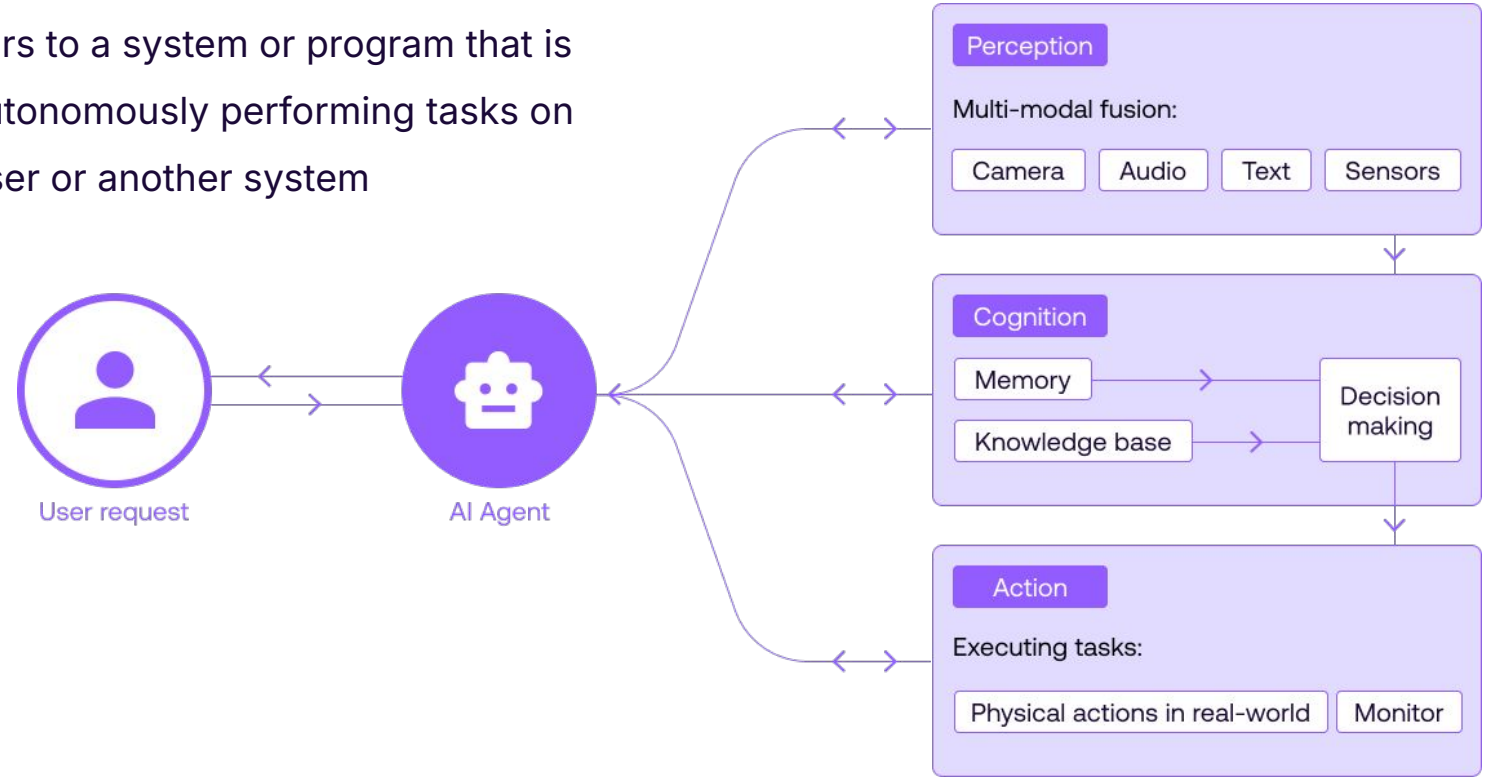


Why micro-agents?

Because, much like **microservices**, they are small units that work together to perform complex tasks. Through effective orchestration & choreography, each micro-agent contributes its specialized function while interacting with others

What are agents?

An agent refers to a system or program that is capable of autonomously performing tasks on behalf of a user or another system

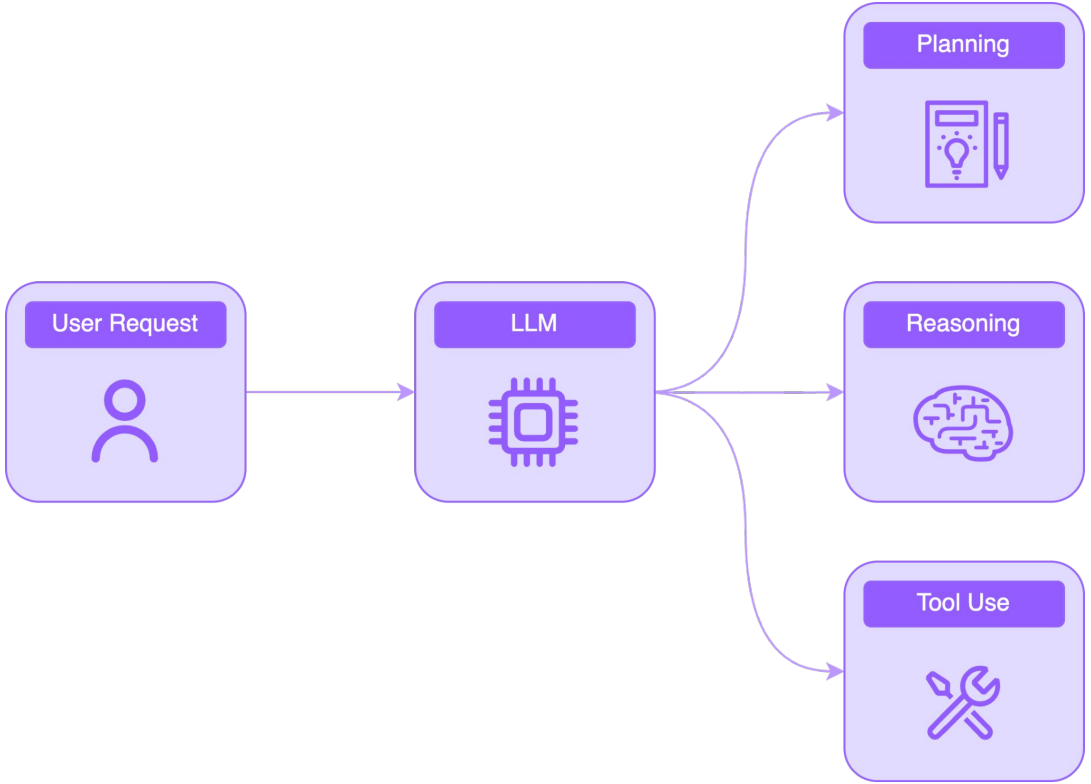


Roombas are agents

- **Perception**: Roombas use sensors (e.g., infrared, cameras, bump sensors) to detect obstacles, dirt, and floor types.
- **Cognition**: Based on sensor inputs, they decide where to move, when to change direction, and when to clean specific areas.
- **Action**: They execute actions like moving, turning, stopping, vacuuming, and docking.
- **Autonomy**: Roombas operate independently. They navigate to their docking station to recharge automatically.

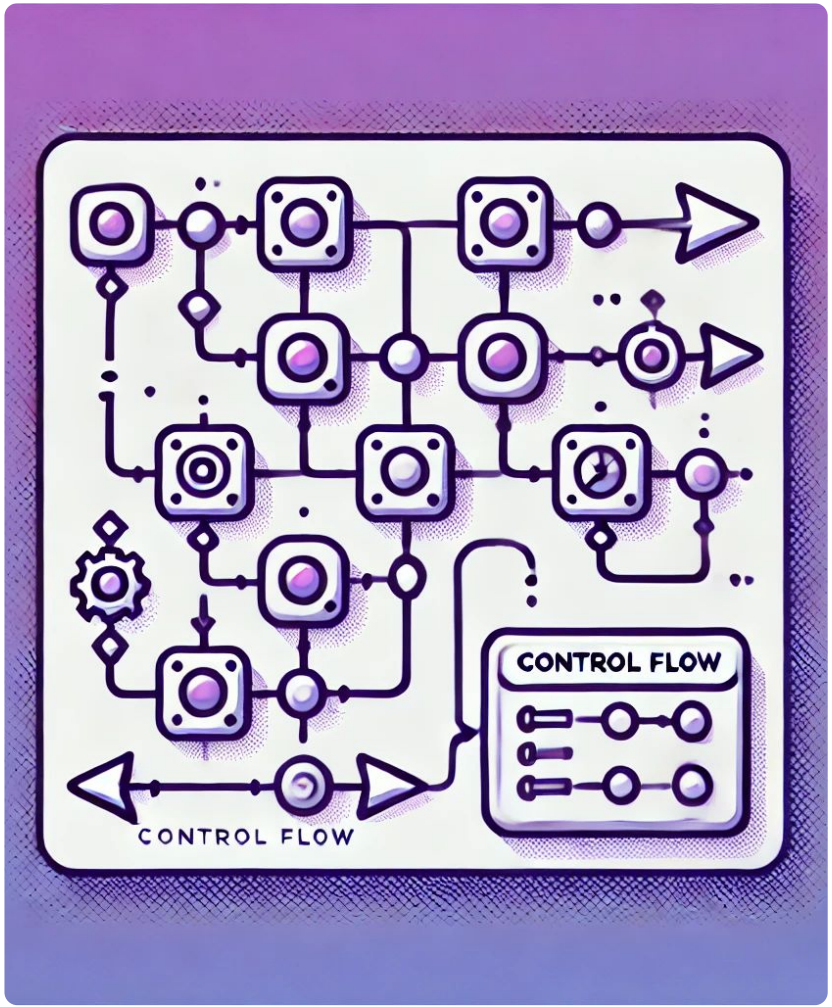
What are agents (today)?

An LLM agent is an AI system that uses a large language model for **reasoning**, **planning**, and **tool use** to autonomously solve tasks, interact with environments, and adapt based on feedback.



What are multi-agent systems (MAS)?

A multi-agent system (MAS) is a network of autonomous agents that interact and collaborate to achieve **complex goals**. These agents can work independently or coordinate with others, making the system more scalable, adaptable, and efficient than a single-agent approach.

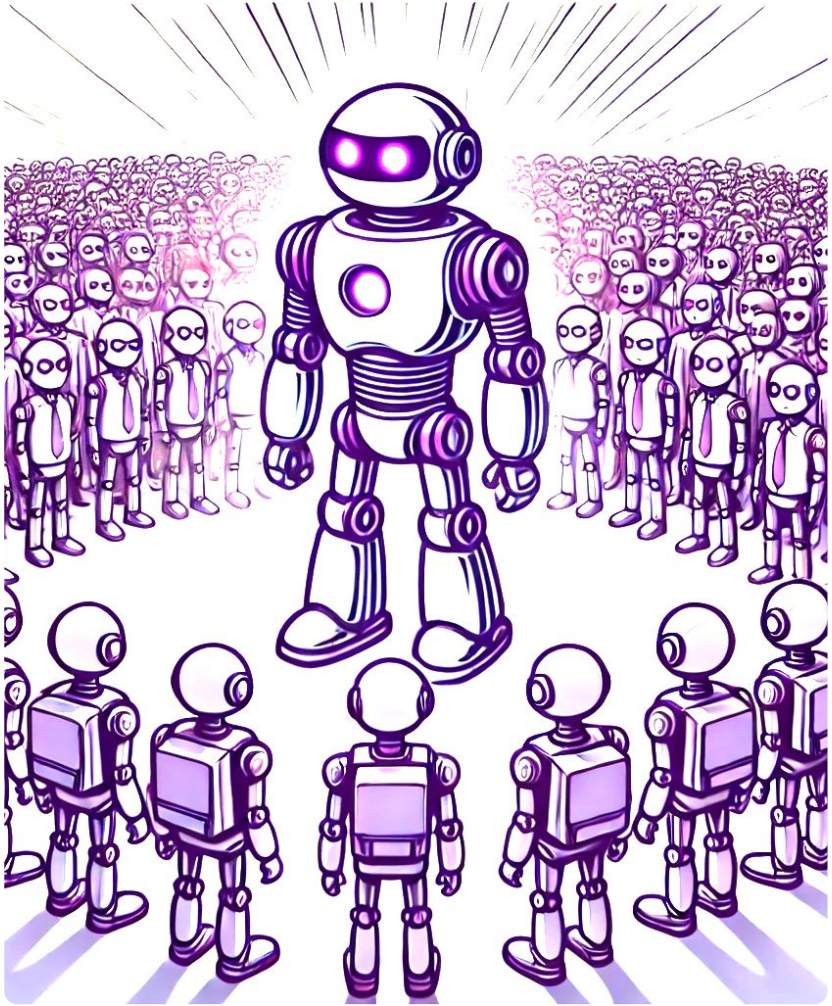


Control Flow

Control flow refers to **how the system coordinates** which agent acts next, how tasks are allocated, and how information is exchanged among agents.

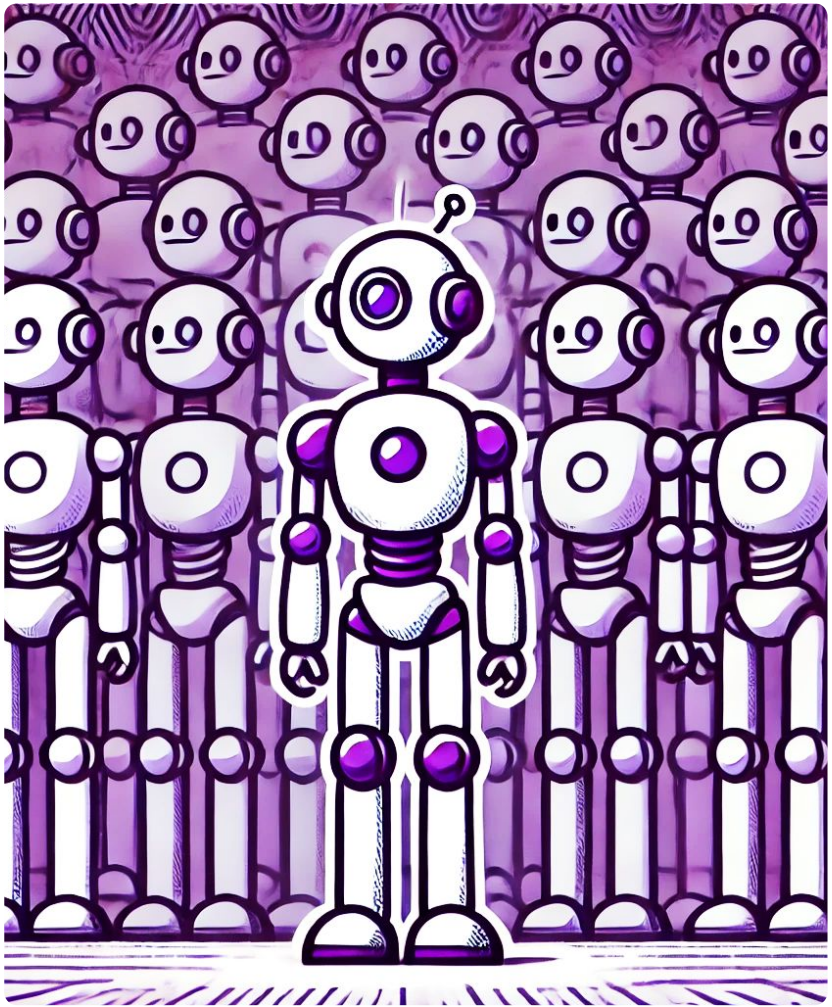
When do we need multi-agents?

- **Reduced Prompt Complexity**: Splitting tasks among multiple agents **avoids** overly long, **fragile prompts** that can confuse a single LLM.
- **Noise Minimization**: Specialized agents handle specific subtasks, **reducing irrelevant context and confusion** compared to a single agent managing many tools.
- **Efficient Parallelism**: Multiple agents can operate **concurrently**, whereas a single agent may struggle with parallel processing.



When do we need multi-agents?

- **Optimized Context Management**: Each agent maintains a smaller, focused context, lowering computational costs and mitigating the issues of **growing context sizes**.
- **Enhanced Specialization & Scalability**: Tailoring agents for specific functions improves performance, allows for **easier updates**, and scales better than a monolithic single-agent system.

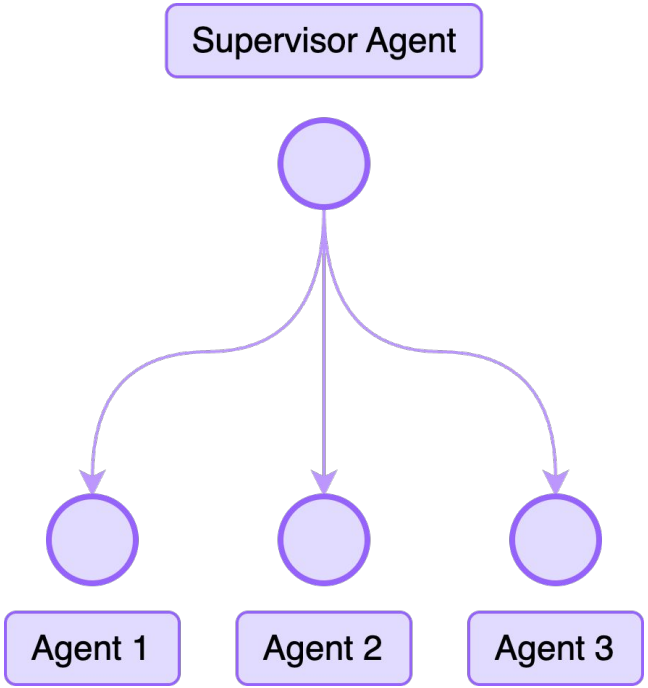


Let’s look at some patterns

Control Flow Patterns in Multi-Agent Systems

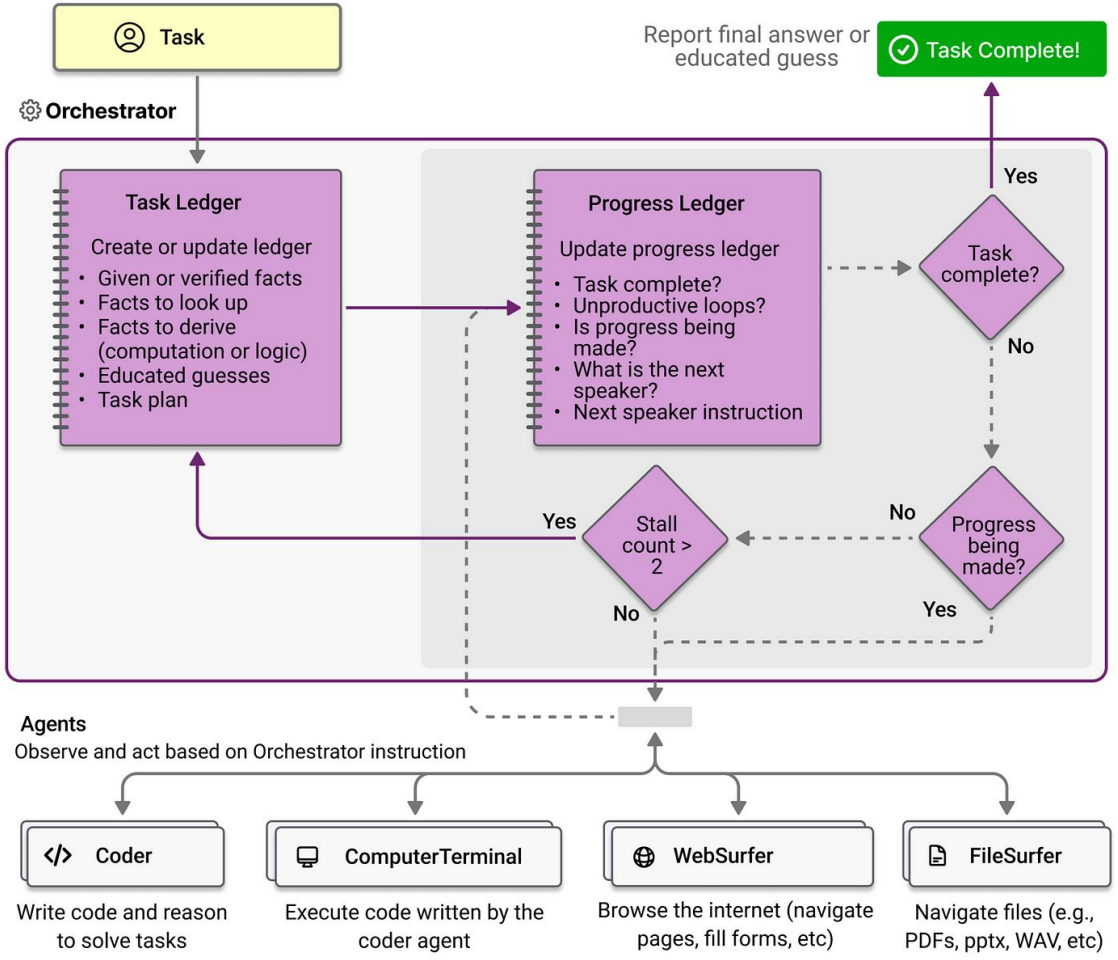
Supervisor

- Features a **deterministic** and **controlled workflow**, often modeled with graphs.
- A central agent **orchestrates subordinate agents** to achieve system objectives.
- Ensures organized task execution and clear command hierarchy.



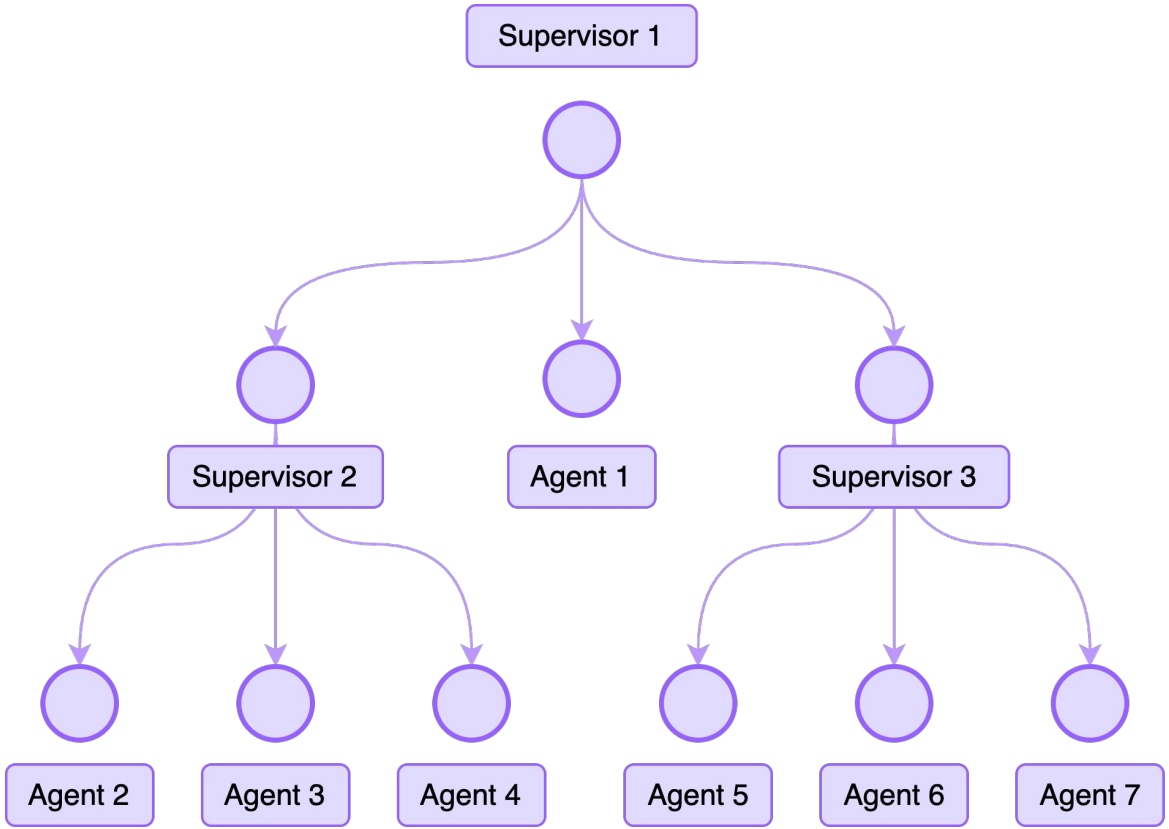
Ex: Microsoft's Magentic-One

- **Ledger Coordination**: The orchestrator uses a task ledger (with facts, guesses, and plans) and a progress ledger to monitor subtask completion.
- **Dynamic Delegation**: It assigns the next subtask to the best-suited agent based on current progress.
- **Adaptive Recovery:** It tracks stalls with a counter and revises plans when progress slows, ensuring robust error recovery.

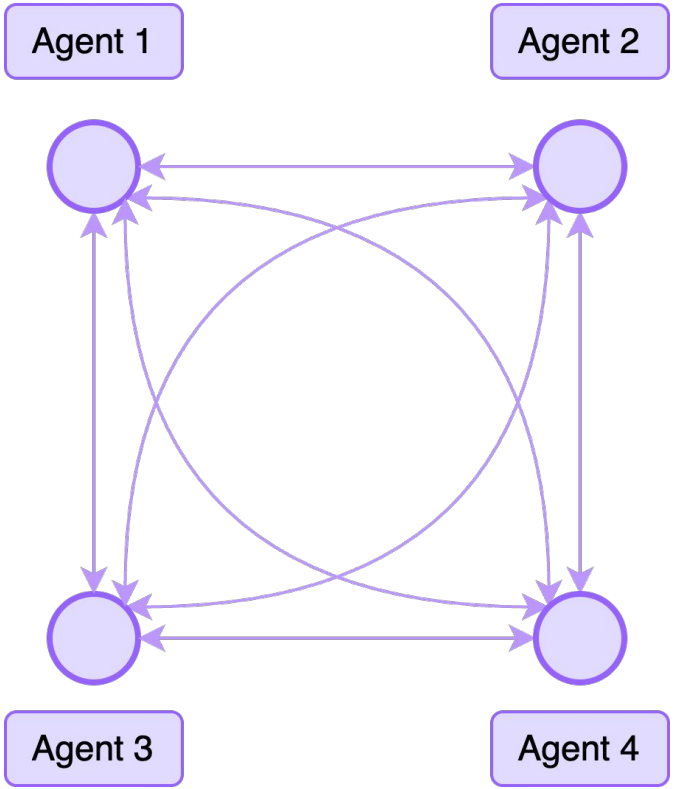


Hierarchical

- Emulates organizational structures with defined **chains of command**.
- Facilitates clear authority levels and streamlined decision-making processes.
- Enhances **scalability** and **manageability** in complex systems.



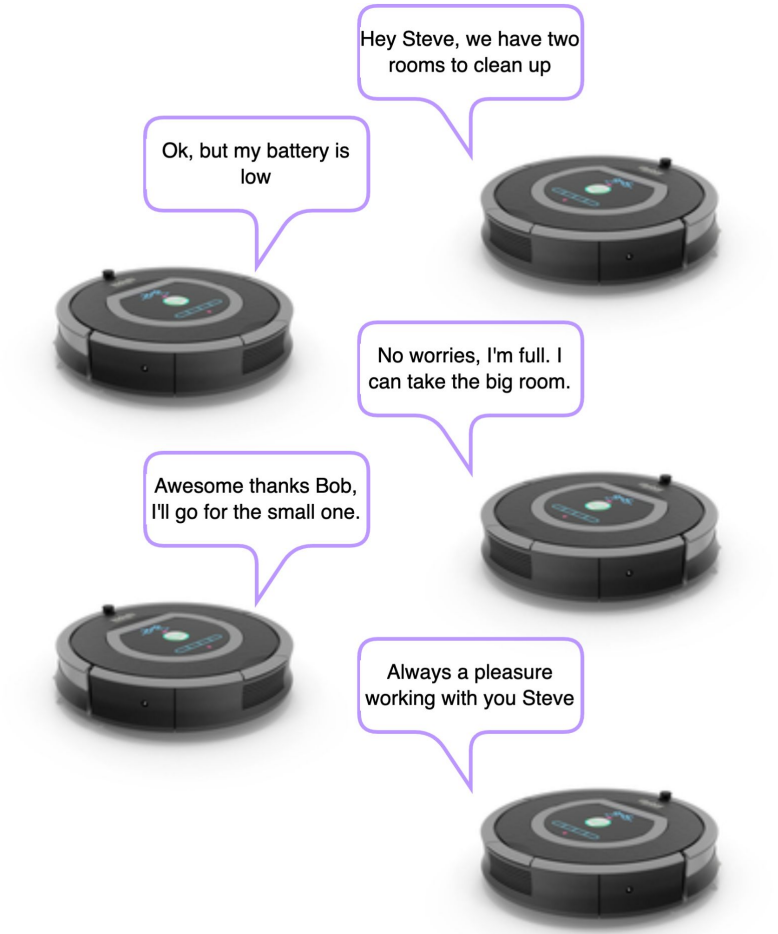
Network



- Exhibits **non-deterministic** interactions suitable for simulations, role-playing, debates, and negotiations.
- Resembles a chat room where agents take turns communicating, reaching consensus to determine subsequent speakers.
- Promotes **dynamic** and **flexible** agent interactions.

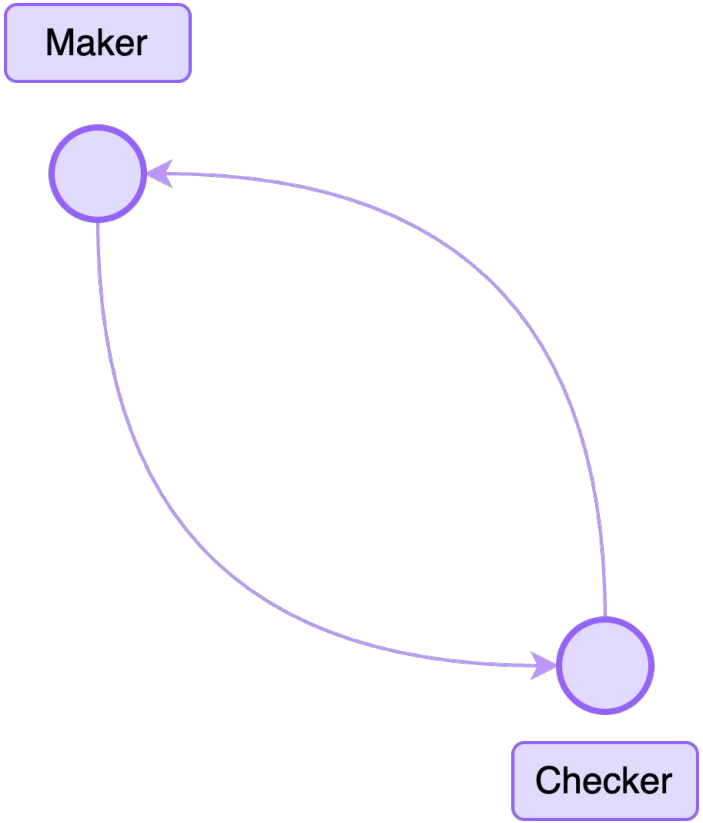
Ex: Roomba Network

Two roombas were tasked to clean 2 rooms. But one room is bigger than the other.



"Bob went on to clean the big room and steve the small room. Bob received a bonus that year."

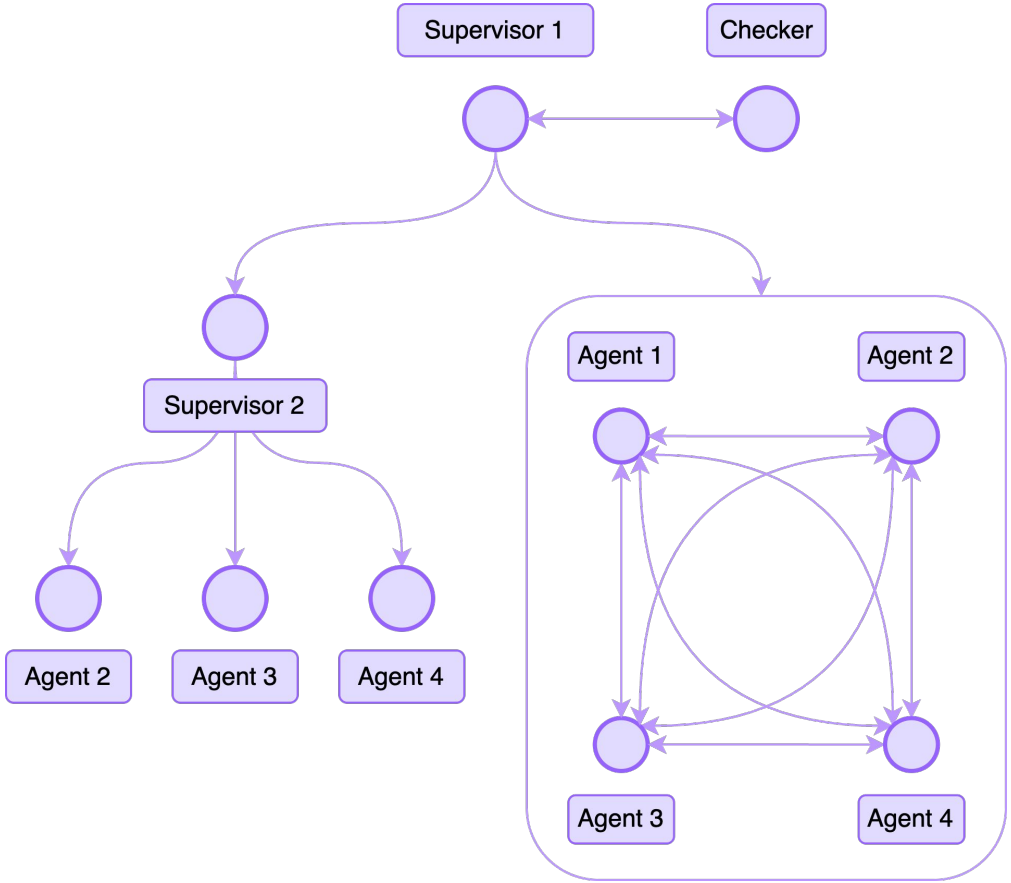
Maker-Checker



- Inspired by financial systems where one agent **verifies** the work of another.
- Enhances system reliability.
- Ensures **accuracy** and **compliance** within agent operations.

Custom

- Combines elements from various patterns to suit specific situational needs.
- Highlights that there is no one-size-fits-all approach; **patterns should be tailored to the context.**
- Encourages flexibility and adaptability in system design.



Communication

How is information passed around?

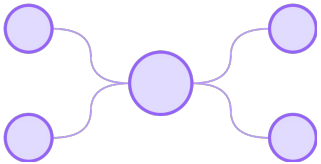
Communication between agents

DM (Point-to-Point Communication)

- **Description**: Agents send messages directly to specific other agents.
- **Usage**: This is common when an agent knows the exact recipient and needs to exchange precise information.

Broadcast Communication

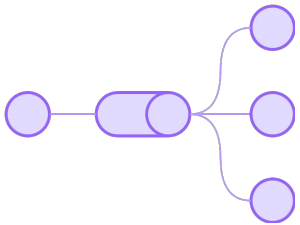
- **Description**: A message is sent from one agent to all other agents (or a predefined group).
- **Usage**: Useful when the sender wants to disseminate information widely.



Communication between agents

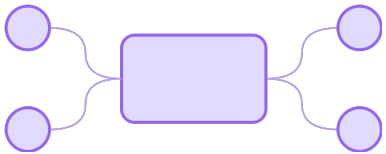
Publish/Subscribe

- **Description**: Agents publish messages to an event bus where other agents subscribe to specific types of messages.
- **Usage**: This decouples the sender and receiver, allowing for dynamic discovery of information.



Blackboard Systems

- **Description**: A shared data repository (blackboard) is used where agents post messages, and others read and update the shared data.
- **Usage**: It supports collaborative problem solving.



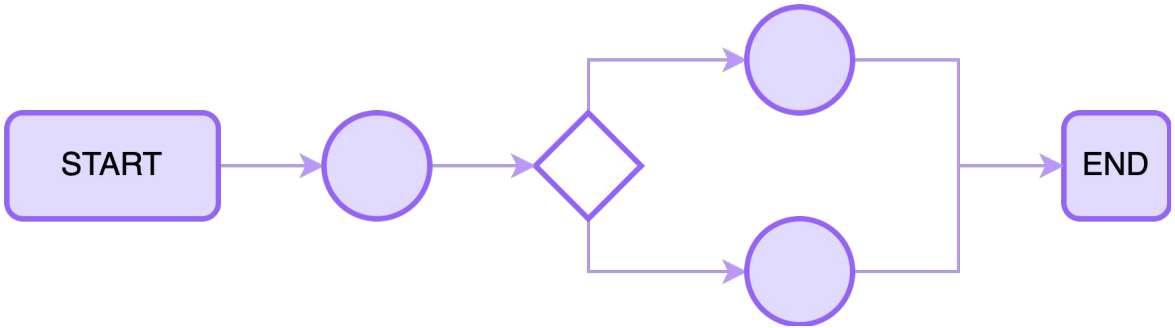
Control Flow Management

Agent management and protocols

Directed Graphs

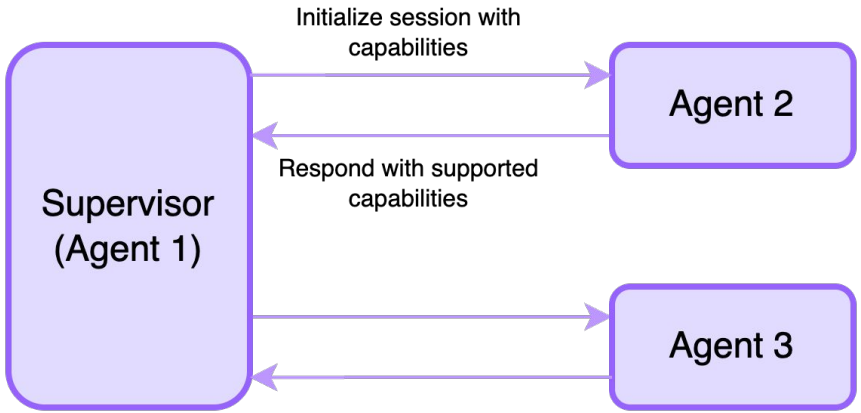
Directed Graphs: Structures where nodes are connected by edges that have a specific direction, representing one-way relationships.

Ex: LangGraph: A framework that uses directed graph structures to model and manage language-related workflows or interactions, often in the context of language processing or AI applications.



Model Context Protocol

The MCP uses a capability-based negotiation system where clients and servers explicitly **declare their supported features during initialization**. Capabilities determine which protocol features and primitives are available during a session.



Contract Net Protocol

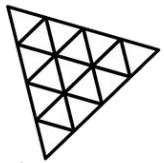
The CNP is a **negotiation-based** coordination mechanism in multi-agent systems.

- **Task Announcement**: One agent (the manager) broadcasts a task or problem.
- **Bidding**: Other agents (contractors) respond with bids or proposals indicating how they can complete the task.
- **Awarding**: The manager evaluates the bids and selects the best proposal, awarding the contract to the winning contractor.

Demo

LangGraph

Langfuse

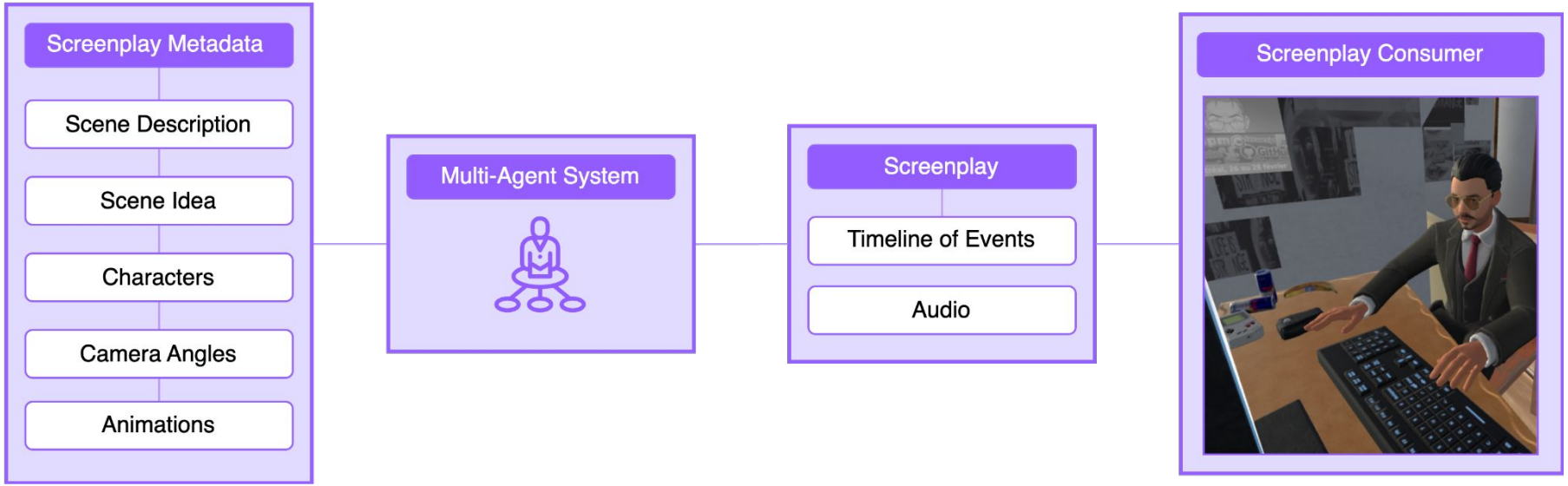


three.js

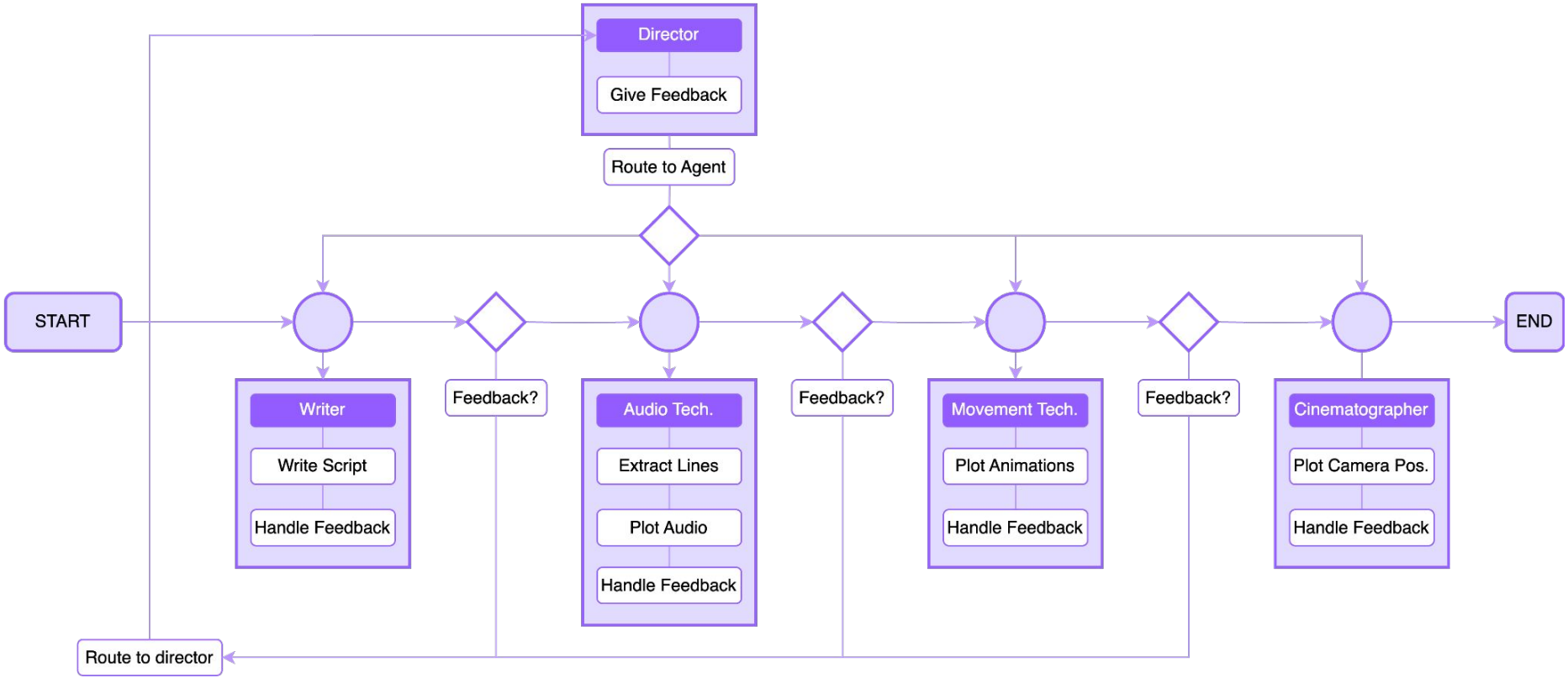
This is Sleazy Ron

- Ron is a crypto analyst
- Ron streams on Twitch to his 6 viewers
- Ron can talk, move, change camera angles
- Most importantly, Ron is a multi-agent system

Behind the Ron



Behind the Ron



```
# Add nodes to the graph
workflow.add_node("director", director_node)
workflow.add_node("writer", writer_node)
workflow.add_node("audio", audio_node)
workflow.add_node("animator", animator_node)
workflow.add_node("cinematographer", cinematographer_node)
workflow.add_node("final", final_node)

# Add edges to the graph
workflow.add_conditional_edges("director", route_to_agent)
workflow.add_conditional_edges("writer", decide_feedback)
workflow.add_conditional_edges("audio", decide_feedback)
workflow.add_conditional_edges("animator", decide_feedback)
workflow.add_conditional_edges("cinematographer", decide_feedback)
workflow.add_edge("final", END)

# Set the entry point
workflow.set_entry_point("writer")

# Compile the graph
return workflow.compile()
```

```python
def decide_feedback(state: WorkflowState) -> str:
    if state["status"] == "requesting_feedback":
        return "director" #move on to director for feedback
    if state["status"] == "starting":
        return state["current_agent"] #move on to next agent

def route_to_agent(state: WorkflowState) -> str:
    return state["current_agent"]
```

```python
# Define the animator node
def animator_node(state: WorkflowState) -> Dict[str, Any]:
    print(f"Animator Status: {state['status']}")

    result = animator.run(state["screenplay"], state["timeline"], state["screenplay_characters"])

    response = {
        **state,
        "agent_output": result["output"],
    }

    if state["status"] == "approved":
        return {
            **response,
            "status": "starting",
            "current_agent": "cinematographer",
        }

    return {
        **response,
        "status": "requesting_feedback",
        "feedback_request": result["feedback_request"],
    }
```

Thank you

Carl Lapierre
Tech Lead

Feedback form

How to contact us

osedea.com
info@osedea.com
LinkedIn