FLORIDA ATLANTIC

COT 6930 - Generative Intelligence
and Software Development Lifecycle

Topic 7 - Multi-Agent Systems Systems

Dr. Fernando Koch
kochf@fau.edu
http://www.fernandokoch.me

Agenda

Foundations of MAS

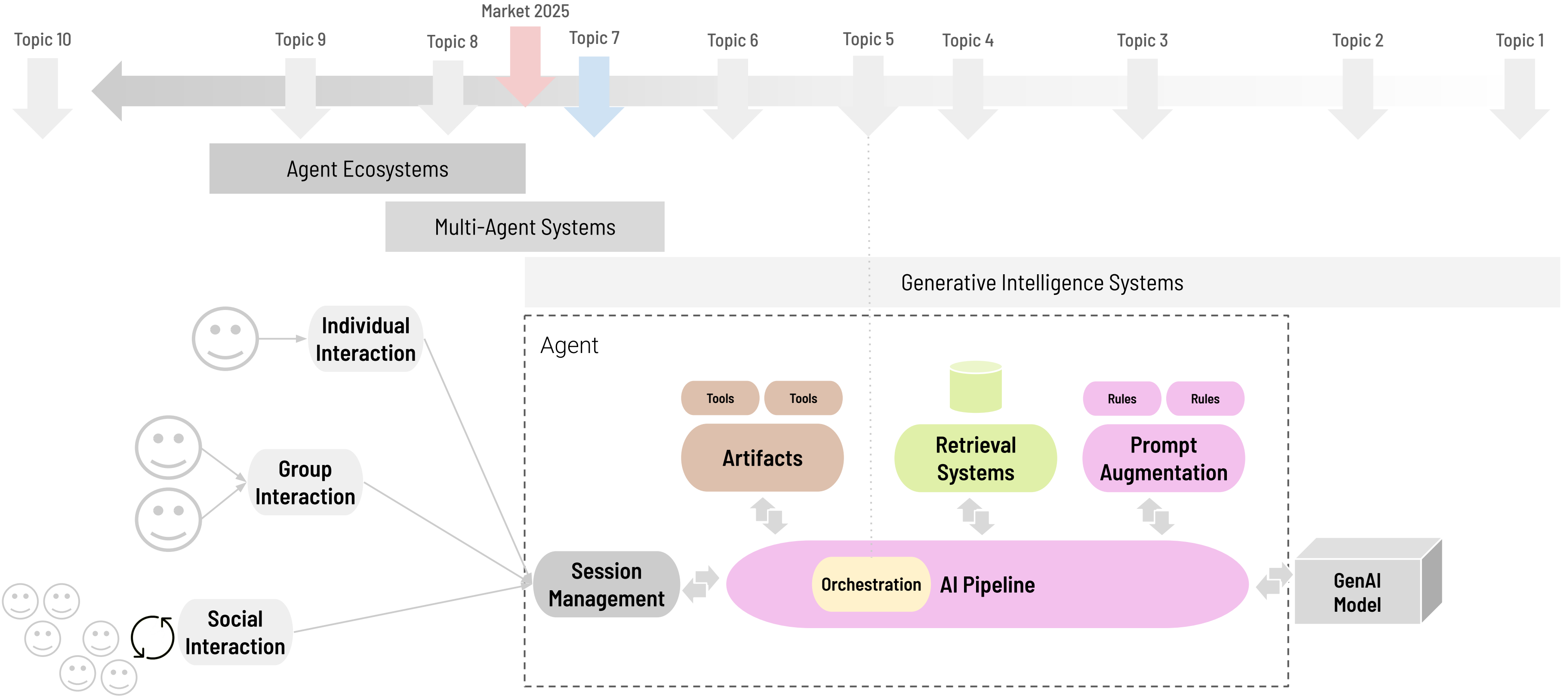
Architectures and Design Patterns

Interaction Protocols and MCP

Our Key Question

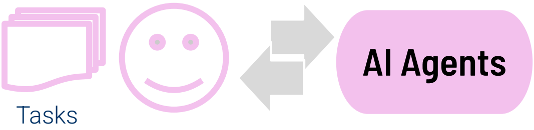
How can we design and coordinate systems composed of multiple interacting agents that collaborate, compete, and adapt within shared environments?

Course Timeline



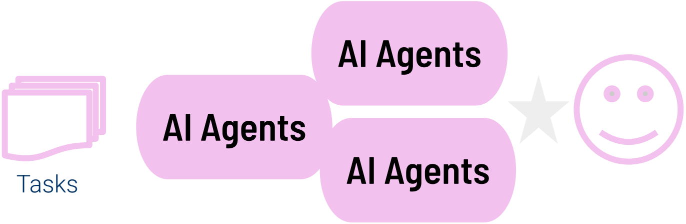
Foundations of MAS

Levels of Automation in Agentic Systems



Level 1 - Human + AI
AI Agent augment human capacity to operate a task

Level 2 - AI + Human
AI Agent implement tasks with Human-In-The-Loop task support

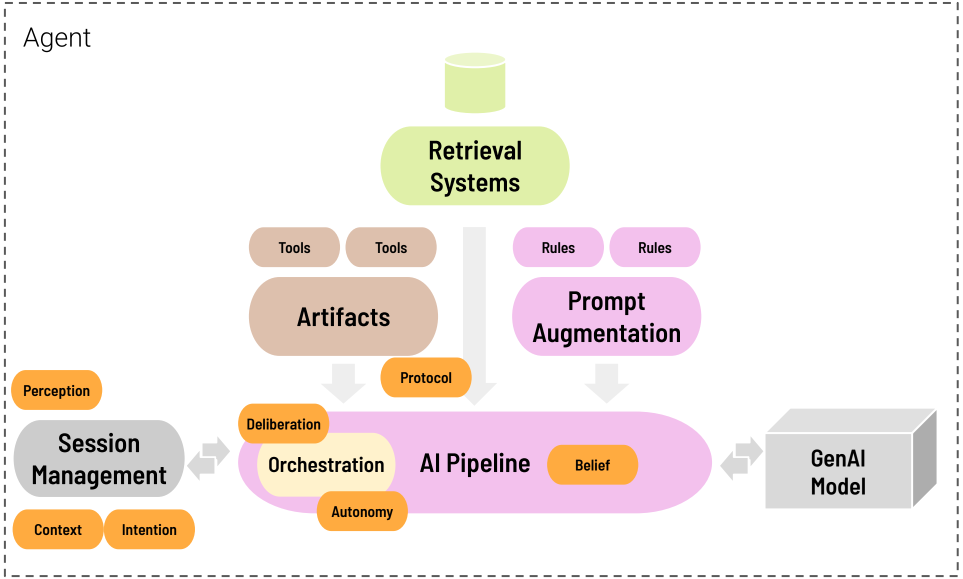


Level 3 - AI + AI w/ Human Oversight
Group of AI Agents implement tasks with Human-In-The-Loop task observation

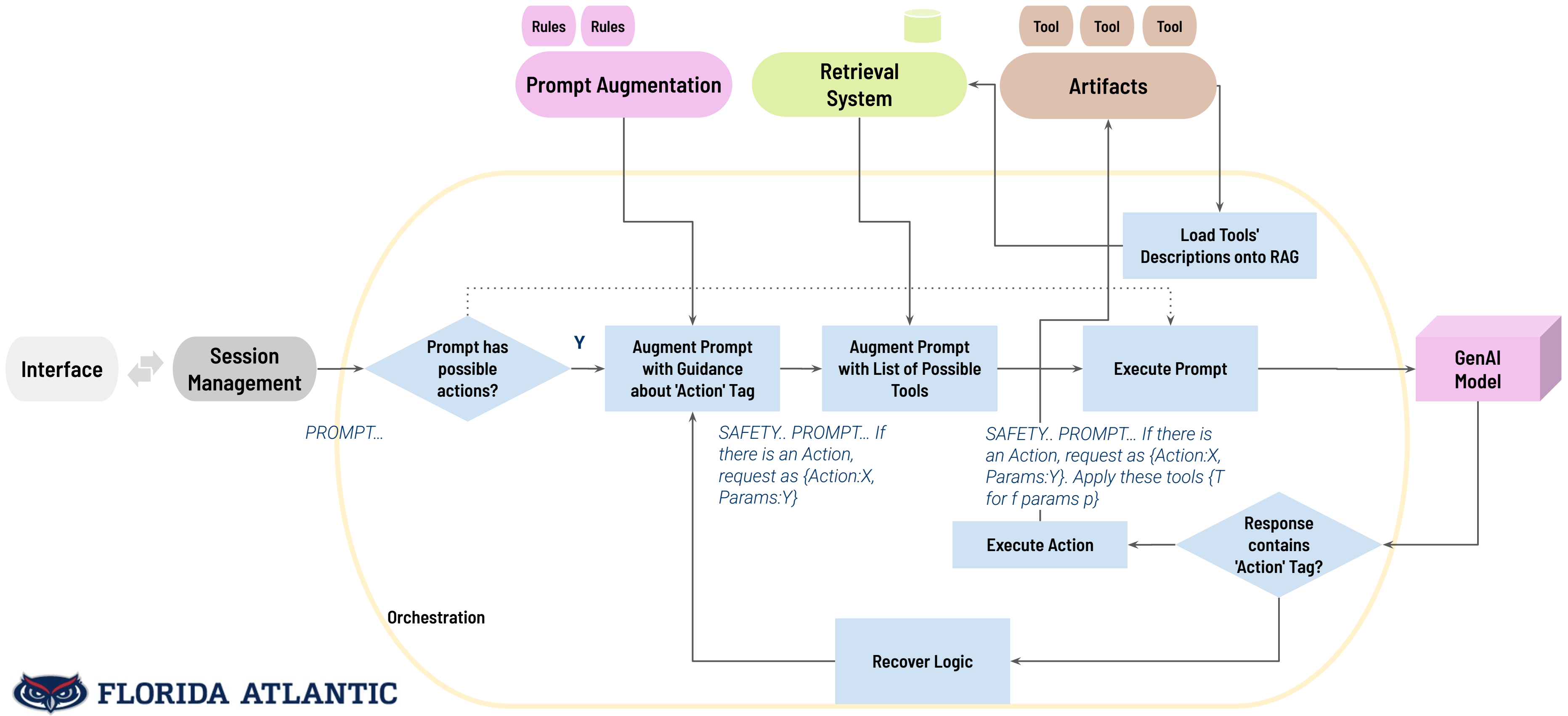
What are Agentic Systems?

Agentic systems =
LLMs + tools + memory + autonomy.

- Perceptions, Beliefs, Intentions.
- Tools
- LLM-driven components capable of planning



Simplified View of ReAct Orchestration Process

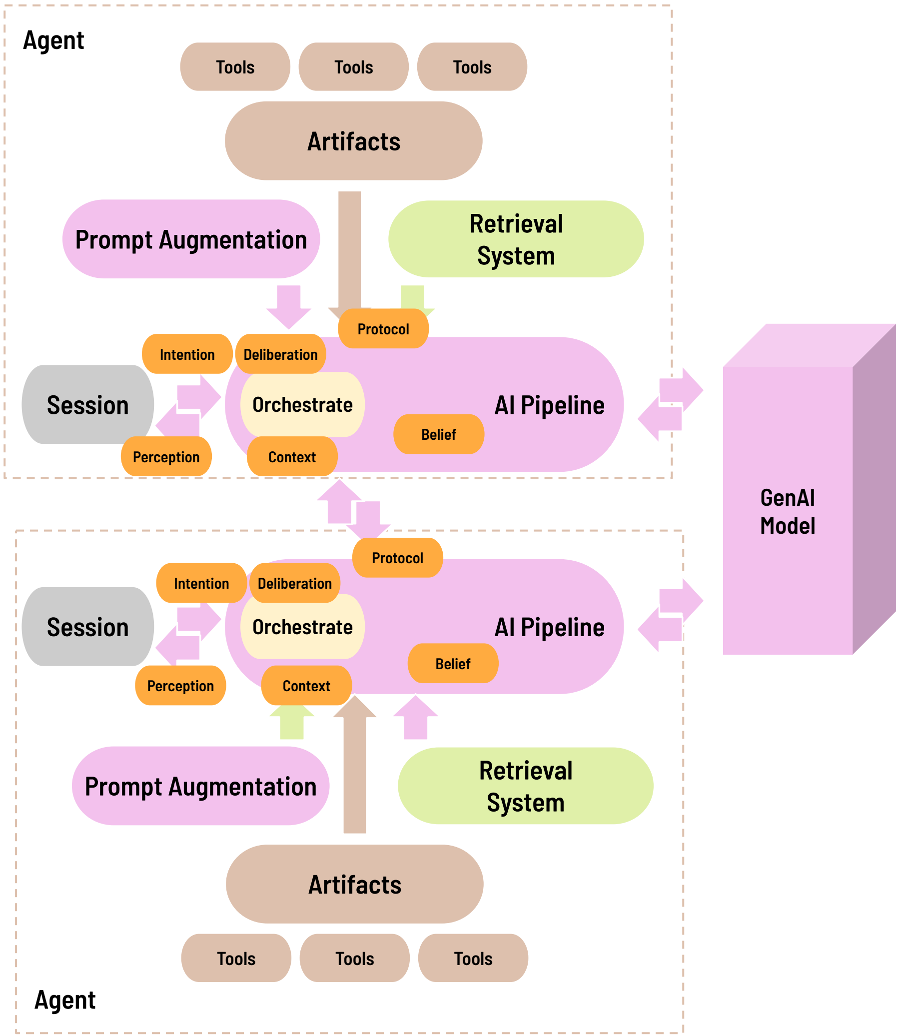


FLORIDA ATLANTIC

What are Multi-Agent Systems?

Cooperative, distributed ecosystems

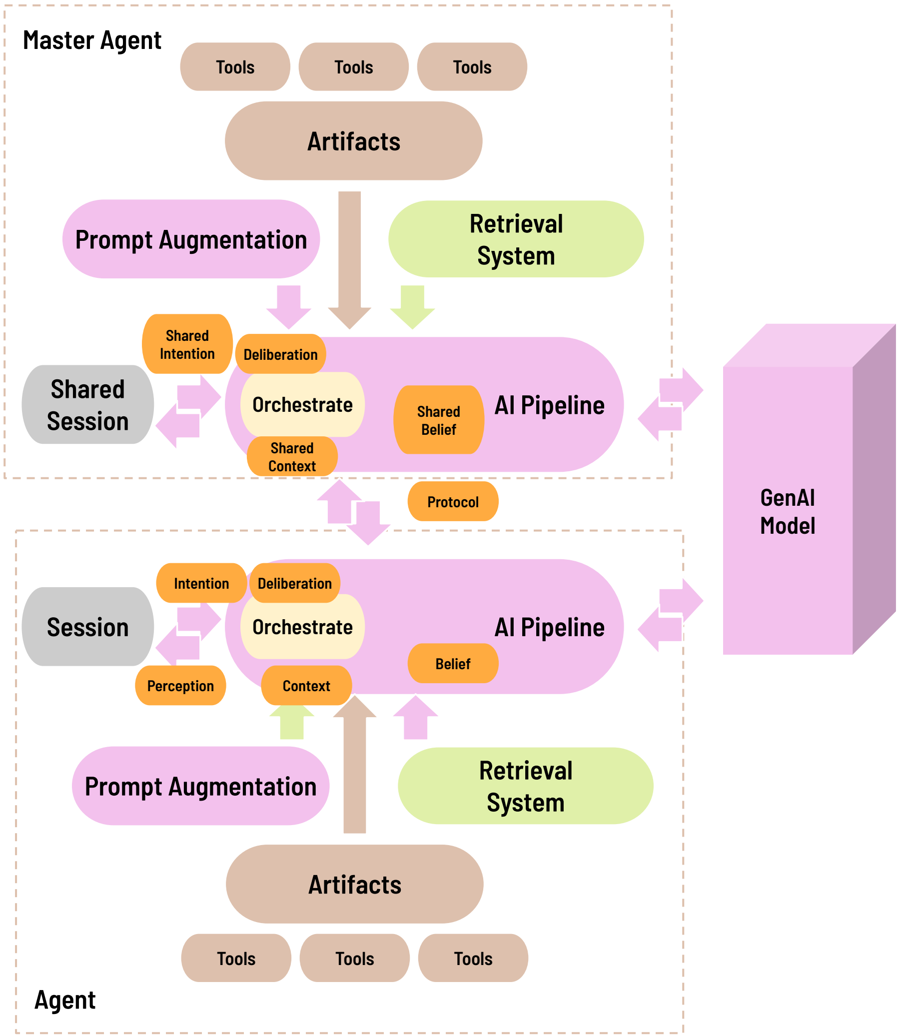
- Act autonomously
- Coordinate toward system-level goals.
- Local context: own beliefs, desires, and intentions (BDI model).
- Adaptive.
- Scalable.



Why do you want Multi-Agent Systems?

Emergent intelligence through collaboration and interaction.

- **Distributed intelligence**.
- **Emergent behavior.**
- Local interactions.
- Resilience.
- Collective problem-solving.

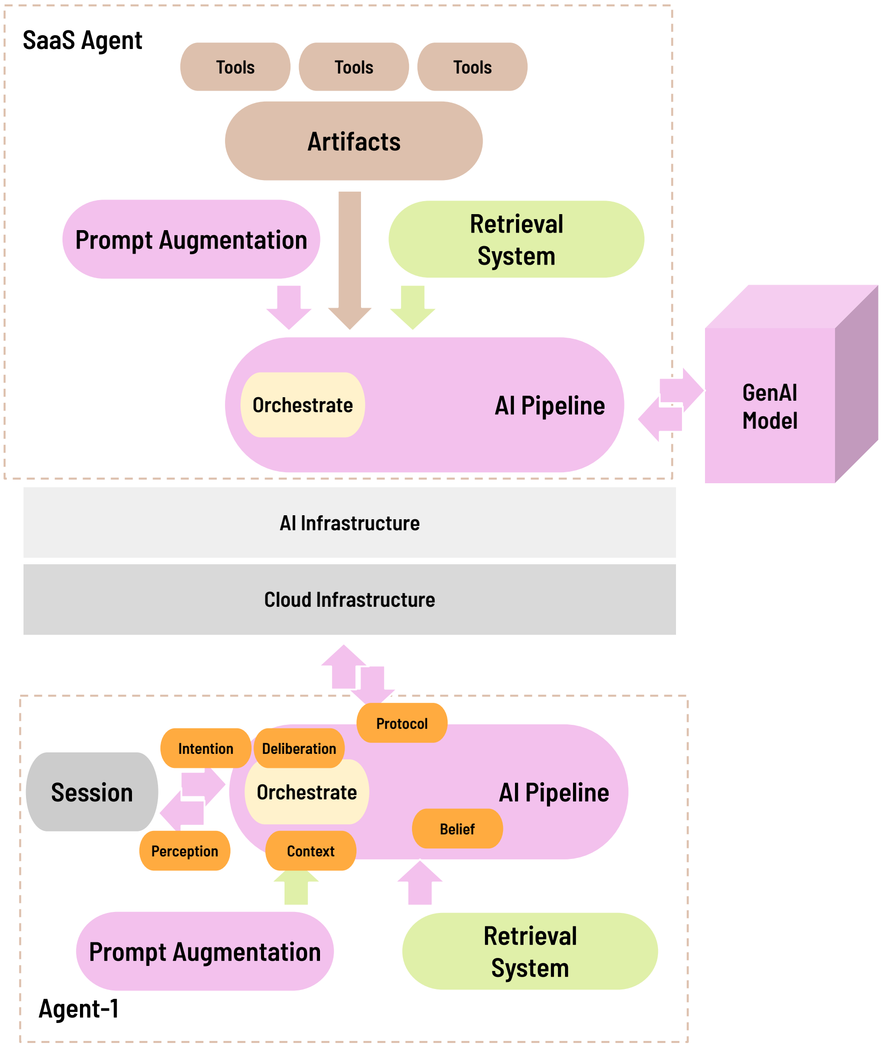


FLORIDA ATLANTIC

Why the Market wants Multi-Agent Systems?

Enterprises are moving workloads to intelligent, **agent-based SaaS 2.0 platforms**

- **MAS as SaaS 2.0**
- **Hyperscaler Infrastructure Needs Intelligent Utilization**
- Cloud Workloads Are Becoming Cognitive
- From Static APIs to Agentic Ecosystems



FLORIDA ATLANTIC

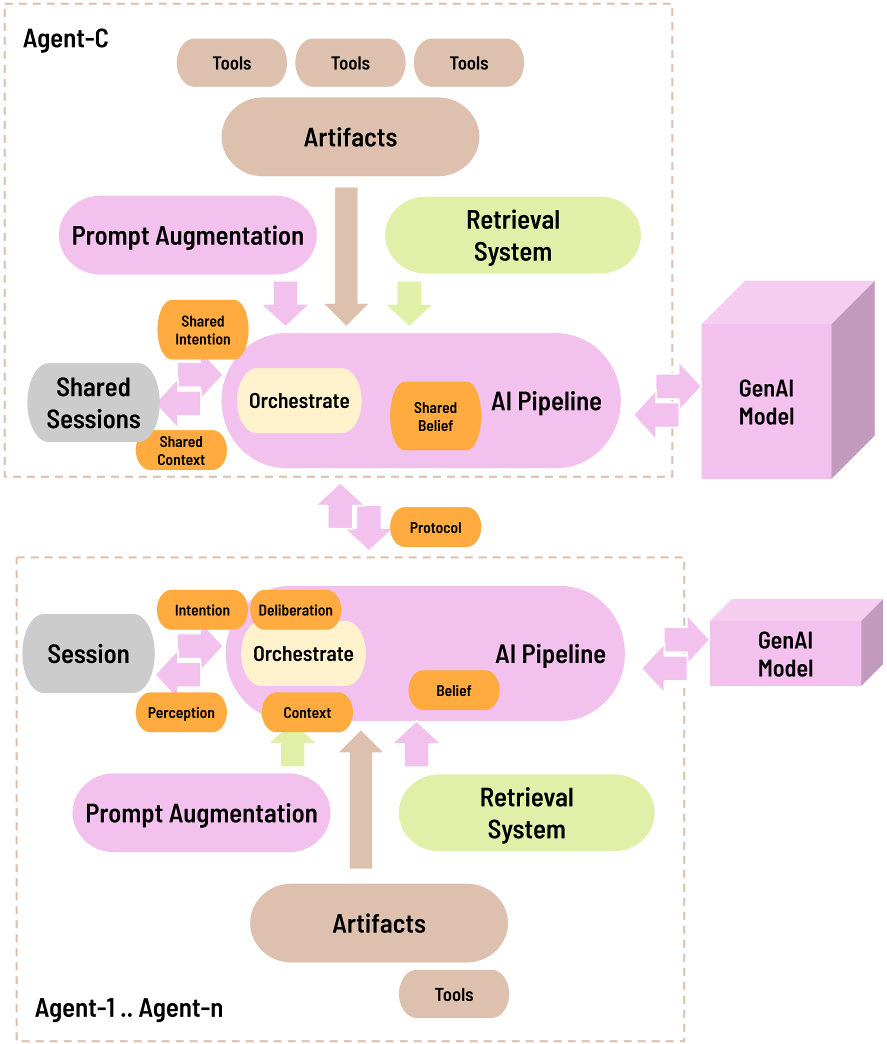
Architectures and Design Patterns

The Simplest Multi-Agent System

Distributed Tool Calling

- Agent-1 needs to execute f*(x)
- Agent-1 does not have the Tools to execute f*(x)
- Agent-1 delegates execution of f*(x) to Agent-C

Agents-C becomes ‘hub of tools’



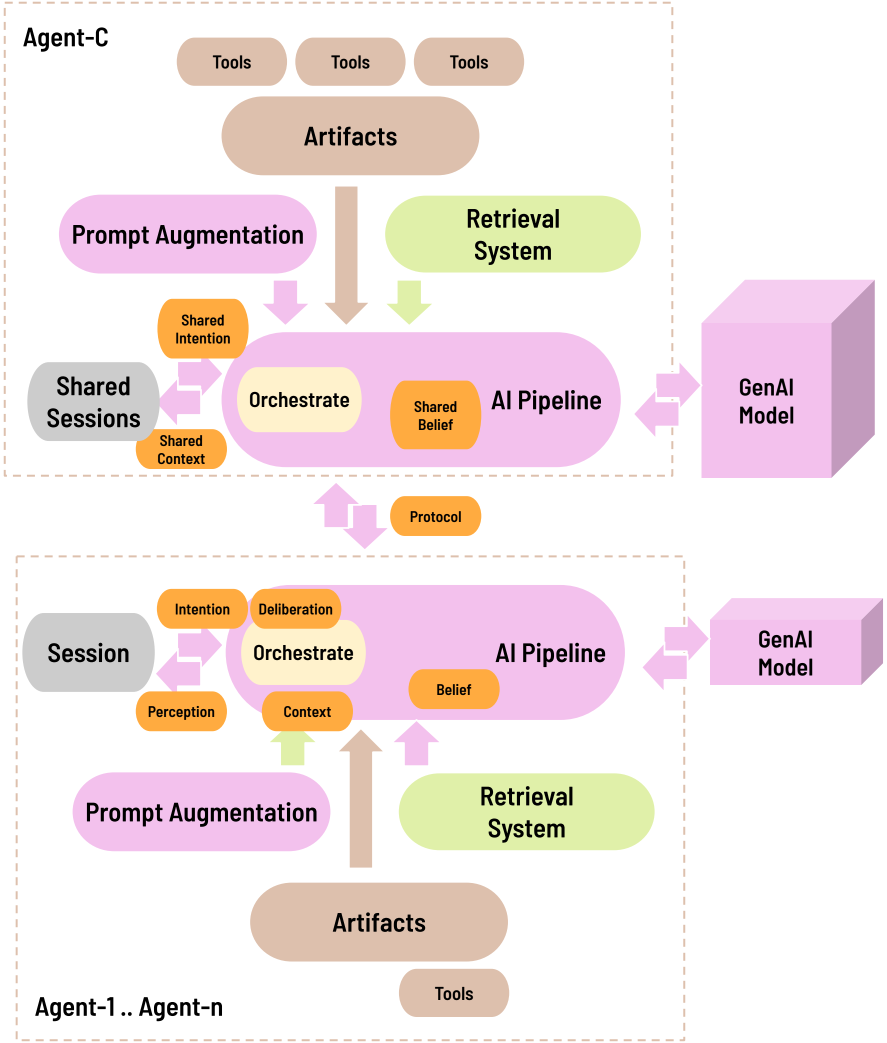
FLORIDA ATLANTIC

The Collective Intelligence Multi-Agent Systems

Shared Knowledge for Collective Action

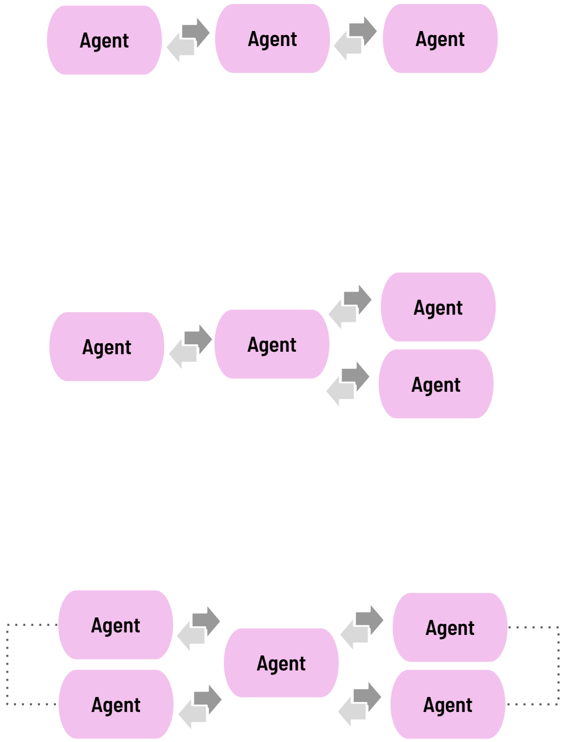
- Agent-1 to Agent-N have local knowledge Kx
- Agent-1 to Agent-N need to share information to execute f*(x)
- Agent-1 to Agent-N upload Kx unto Agent-C, which combines all knowledge in Shared Belief SB
- Agent-1 requests Agent-C to execute f*(x), where $x \subseteq SB + Kx$

Central Agent becomes hubs of knowledge and tools



FLORIDA ATLANTIC

Architectural Patterns



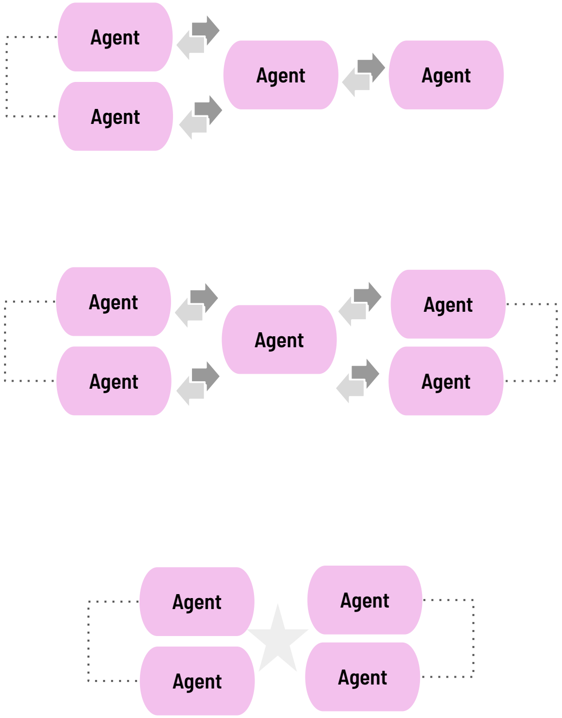
Common patterns guide coordination and intelligence distribution.

- Hierarchical vs. peer-to-peer control.
- Blackboard and mediator architectures.
- Market-based and swarm intelligence patterns.
- Event-driven and goal-oriented coordination.

Design Principles of MAS

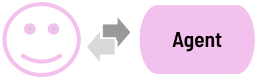
Mirrors organizational and social structures.

- **Modularity**: Agents are self-contained, reusable units.
- **Openness**: Agents join/leave without disrupting the system.
- **Robustness**: Distributed control reduces single points of failure.
- **Scalability**: Coordination grows with agent population.

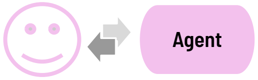


Human-Agent Interactions

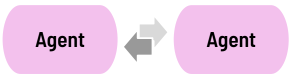
1-N Multi-Agents



Initiated by human

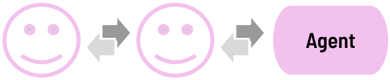


initiated by agent



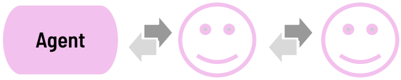
initiated by one agent

1-N Social Intelligence

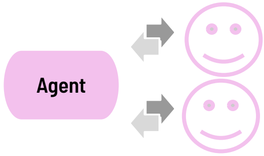


motivated by another human

intermediated by agent

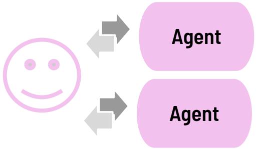


motivated by agent



agent motivates N-humans

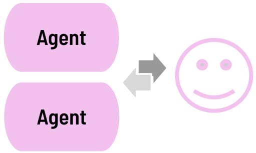
1-N Human / Multi-Agent



initiated by human

motivated by one agent

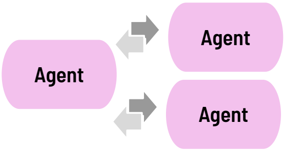
motivated by another agent



initiated by N-agents

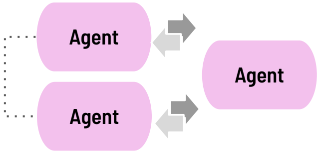
Multi-agent Compositions

1-N Multi-Agent

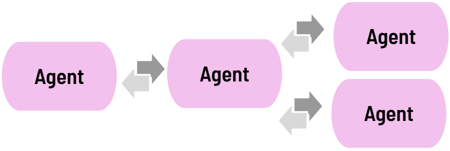


initiated by one agent

motivated by another agent

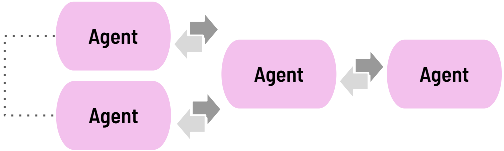


initiated by N-agents

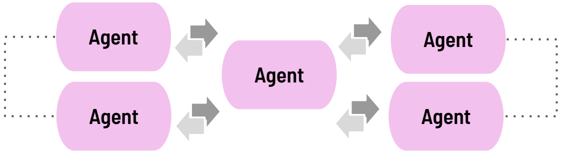


Intermediated by one agent

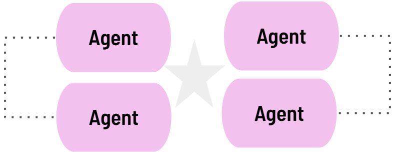
N-M Multi-Agent



initiated by N-agents,
intermediated by another agent



initiated by N-agents,
influencing another M-agents



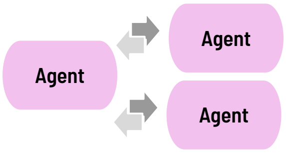
networked interactions

1-N initiated by one agent

A single lead agent delegates specialized subtasks to multiple supporting agents.

Example: Product Development Orchestrator → Multiple Specialized Agents

- A “Planner” agent decomposes an MVP requirement into design, coding, and testing subtasks.
- Each sub-agent (Designer, Coder, Tester) executes independently.
- The orchestrator collects progress and integrates results.
- Enables parallel, modular execution within AI-driven DevOps.



initiated by one agent

1-N motivated by Another Agent

An agent triggers reactions or extensions of reasoning in others.

Example: Customer Query Bot → Chain of Expert Agents

- A Support Agent receives a customer request.
- It activates Billing, Technical, and Policy agents for deeper responses.
- Each contributes insights or resolutions back to the root agent.
- Mimics "chain-of-thought" delegation across AI specialists.

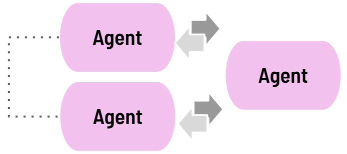
motivated by another agent

N-1 initiated by N Agents

Multiple agents collaborate or compete to influence one central agent's decision.

Example: Market Intelligence Agents → Strategic Decision Agent

- Independent agents monitor financial data, social media, and product trends.
- They submit their findings to a "Decision Synthesizer."
- The central agent evaluates, weighs evidence, and formulates a market response.
- Represents data fusion and consensus-building architectures.



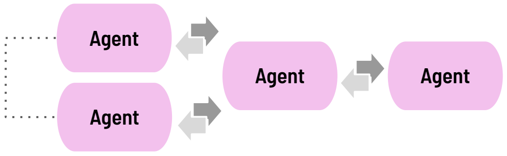
initiated by N-agents

N-N Intermediated by One Agent

Multiple agents interact through an intermediary that coordinates communication.

Example: Supply Chain Agents ↔ Logistics Hub Agent

- Suppliers, transporters, and retailers operate as autonomous agents.
- A Logistics Hub orchestrator mediates information flow and task allocation.
- Balances inventory, shipment timing, and resource allocation.
- Reflects real-time distributed coordination in smart logistics.



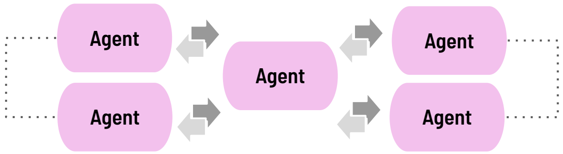
initiated by N-agents,
intermediated by another agent

N-M influencing other M-Agents

Groups of agents collaborate or compete, influencing another group's outcomes.

Example: Adversarial Simulation (Red Team vs. Blue Team Agents)

- Red Team agents simulate cyberattacks; Blue Team agents defend.
- Both sides learn dynamically, improving strategies iteratively.
- The system evolves resilience through adversarial reinforcement.
- Useful in cybersecurity, policy testing, or autonomous system safety.



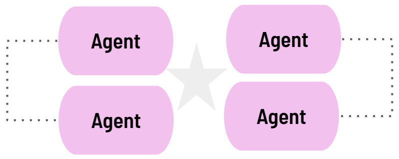
initiated by N-agents,
influencing another M-agents

N-M Networked Interactions

Agents operate as a web of interdependent peers without a single coordinator.

Example: AI Collaboration Network for Research & Innovation

- Agents representing different research labs share hypotheses, results, and data.
- Each node both consumes and produces knowledge.
- Emergent behaviors surface from global collaboration.
- This models open, self-organizing AI ecosystems!



networked interactions

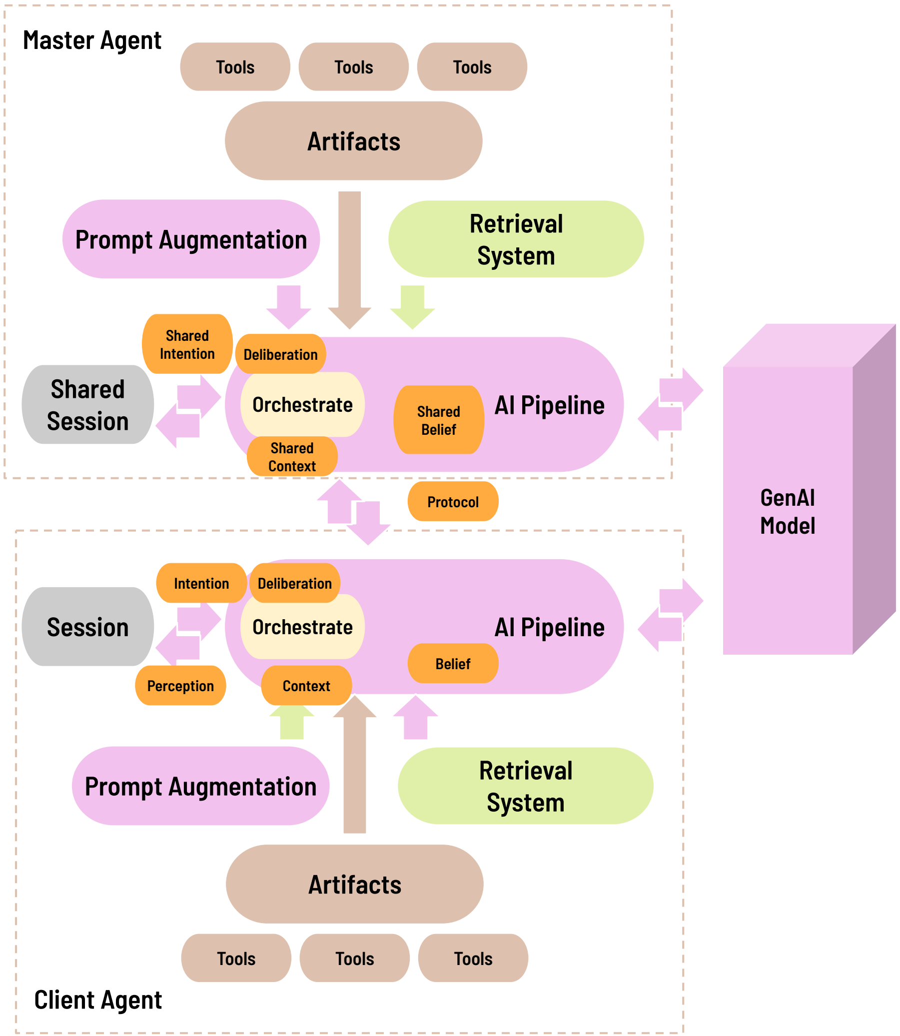
The foundation for agentic Web 4.0 !!!

Interaction Protocols and MCP

Interaction Types

Agents interact through **cooperation**, **competition**, or **coordination**.

- **Cooperative**: shared goals, collective success.
- **Competitive**: conflicting objectives drive innovation.
- **Coordinative**: task alignment and role negotiation.
- **Hybrid** interactions evolve dynamically.

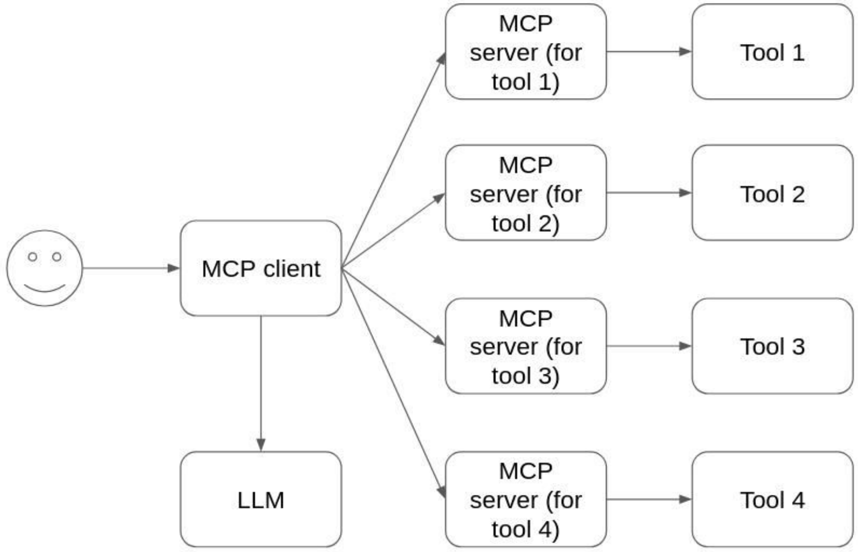


FLORIDA ATLANTIC

Model Context Protocol (MCP)

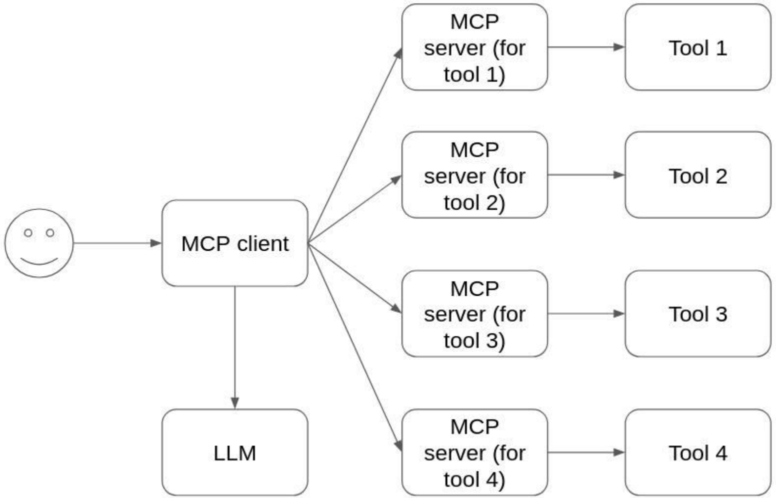
Defines a standard protocol for AI models and agents to share tools, context, and memory securely.

- Developed to make AI systems interoperable across models (e.g., OpenAI, Anthropic).
- Connects LLMs with external systems through standardized context exchange.
- Agents can dynamically “mount” capabilities like APIs, databases, or services.
- Foundation for cross-agent collaboration and secure tool usage.

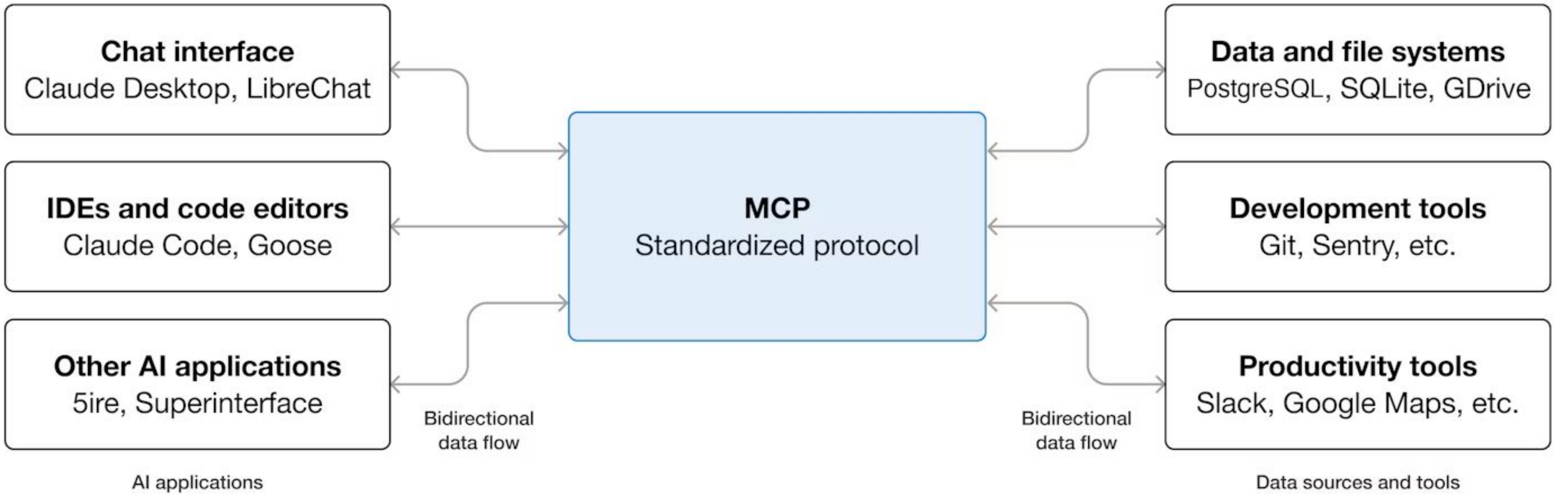


Model Context Protocol (MCP)

Component	Role	Examples / Notes
MCP Client	The AI/agent side that requests context, tools, or data to support reasoning and actions.	E.g. a Claude-based assistant, LangChain agent, a local AI app integrating with external systems. Microsoft Learn +4
MCP Server	Exposes specific data sources, tools, or services through the MCP protocol so that agents can access them.	E.g. servers that wrap databases, GitHub, Slack, file systems, or custom enterprise systems. Model Context Pr... +4
Protocol / Transport Layer	The communication medium and format (messages, RPCs) between clients and servers.	MCP uses JSON-RPC 2.0 over stdio or HTTP, sometimes with SSE (Server-Sent Events) transport. Wikipedia +3
Schema & Specification	Formal definitions for messages, operations, tool interfaces, and context management.	The official MCP spec defines the request/response schemas, tool metadata, error handling, etc. Model Context Pr... +1



Model Context Protocol (MCP)



MCP Server

https://modelcontextprotocol.io/docs/develop/build-server

Develop with MCP

Build an MCP server

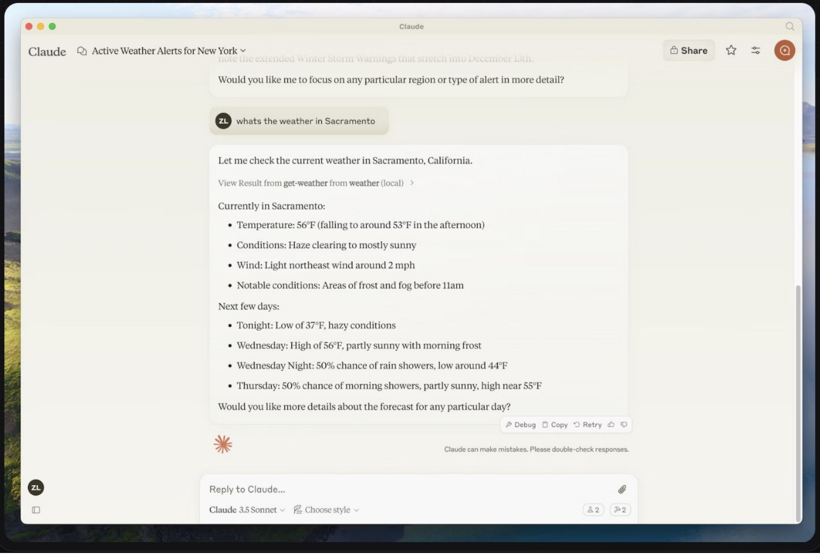
Copy page

Get started building your own server to use in Claude for Desktop and other clients.

In this tutorial, we'll build a simple MCP weather server and connect it to a host, Claude for Desktop.

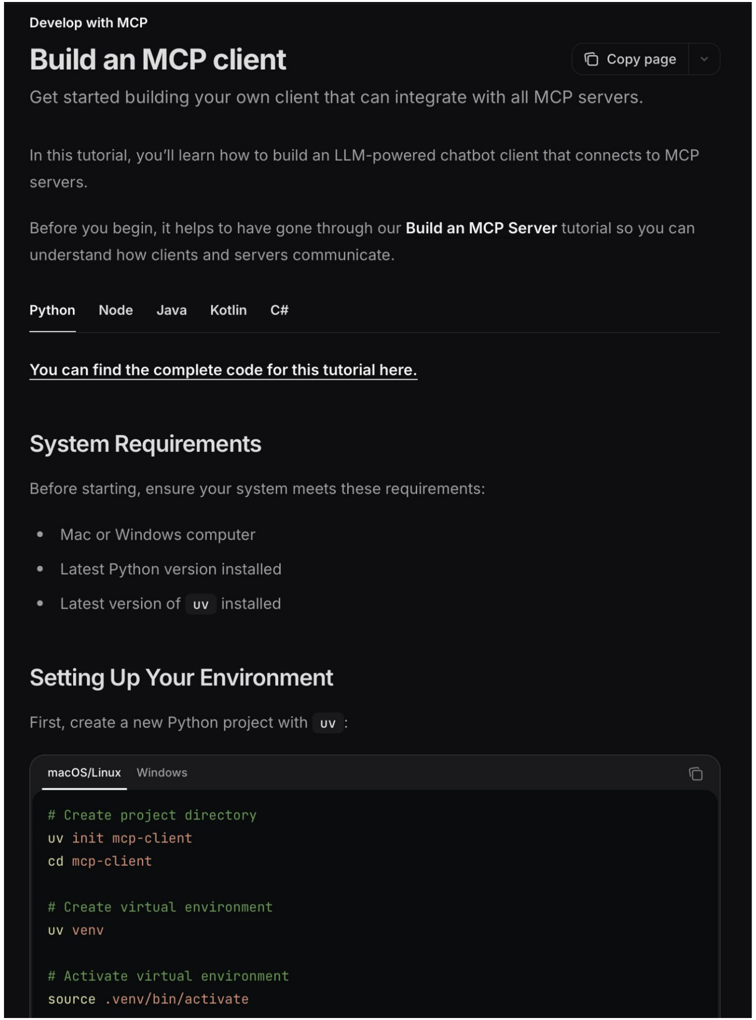
What we'll be building

We'll build a server that exposes two tools: `get_alerts` and `get_forecast`. Then we'll connect the server to an MCP host (in this case, Claude for Desktop):



FLORIDA ATLANTIC

MCP Client



Develop with MCP

Build an MCP client

Copy page

Get started building your own client that can integrate with all MCP servers.

In this tutorial, you'll learn how to build an LLM-powered chatbot client that connects to MCP servers.

Before you begin, it helps to have gone through our **Build an MCP Server** tutorial so you can understand how clients and servers communicate.

Python Node Java Kotlin C#

You can find the complete code for this tutorial here.

System Requirements

Before starting, ensure your system meets these requirements:

- Mac or Windows computer
- Latest Python version installed
- Latest version of `uv` installed

Setting Up Your Environment

First, create a new Python project with `uv`:

macOS/Linux Windows

```
# Create project directory
uv init mcp-client
cd mcp-client

# Create virtual environment
uv venv

# Activate virtual environment
source .venv/bin/activate
```

https://modelcontextprotocol.io/docs/develop/build-client

Connect to Remote MCP Server

https://modelcontextprotocol.io/docs/develop/connect-remote-servers

Develop with MCP

Connect to remote MCP Servers

Copy page

Learn how to connect Claude to remote MCP servers and extend its capabilities with internet-hosted tools and data sources

Remote MCP servers extend AI applications' capabilities beyond your local environment, providing access to internet-hosted tools, services, and data sources. By connecting to remote MCP servers, you transform AI assistants from helpful tools into informed teammates capable of handling complex, multi-step projects with real-time access to external resources.

Many clients now support remote MCP servers, enabling a wide range of integration possibilities. This guide demonstrates how to connect to remote MCP servers using **Claude** as an example, one of the **many clients that support MCP**. While we focus on Claude's implementation through Custom Connectors, the concepts apply broadly to other MCP-compatible clients.

Understanding Remote MCP Servers

Remote MCP servers function similarly to local MCP servers but are hosted on the internet rather than your local machine. They expose tools, prompts, and resources that Claude can use to perform tasks on your behalf. These servers can integrate with various services such as project management tools, documentation systems, code repositories, and any other API-enabled service.

The key advantage of remote MCP servers is their accessibility. Unlike local servers that require installation and configuration on each device, remote servers are available from any MCP client with an internet connection. This makes them ideal for web-based AI applications, integrations that emphasize ease-of-use and services that require server-side processing or authentication.

What are Custom Connectors?

Custom Connectors serve as the bridge between Claude and remote MCP servers. They allow you to connect Claude directly to the tools and data sources that matter most to your workflows, enabling Claude to operate within your favorite software and draw insights from the complete context of your external tools.

With Custom Connectors, you can:

EXERCISE

Exercise 7 - Multi-Agent Experiment

Objective:

Develop a basic Agentic AI using MCP architecture.

Go to Canvas -> Assignments

COT 6930 - Generative Intelligence and Software Development Lifecycles

Dr. Fernando Koch
kochf@fau.edu
http://www.fernandokoch.me