Scalable Multi-Agent AI with AutoGen

Udaiappa Ramachandran (Udai)

CTO/CSO @ Akumina Inc., | Microsoft MVP (AI)

Web: https://udai.io

LinkedIn: https://linkedin.com/in/udair

Meetup: https://meetup.com/nashuaug

Boston Code Camp 38 *Thanks to our Sponsors!*

Microsoft

Platinum

MILL5

PULSAR SECURITY

Gold

slalom

Progress Telerik

Silver

Duende.

Bronze

In-Kind Donations

CALL FOR PAPERS POWERED BY sessionize

Agenda

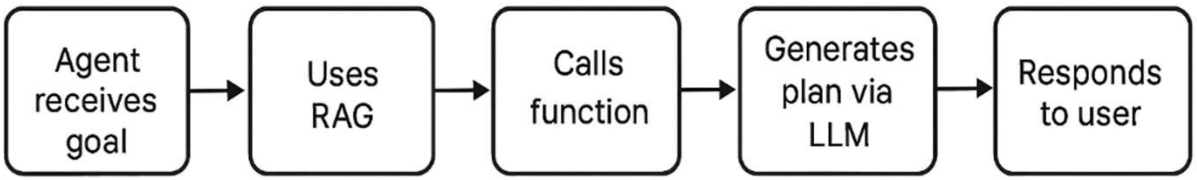
- What Are AI Agents?
  - Understanding the core concept and characteristics
- Building Blocks of Modern AI Systems
  - How LLMs, tools, memory, and chat work together
- Agent Frameworks & AutoGen
  - Overview of frameworks with a deep dive into AutoGen
- Model Context Protocol (MCP)
  - How agents interact with tools, APIs, and environments
- Live Demo & Use Cases
  - Real-world agent conversations and applications

Building Blocks of Modern AI Systems

- Completion
  - Predicts and completes text based on a prompt. It's the core of how LLMs work.
- Chat
  - Enables multi-turn conversations, maintaining context and role-based replies.
- LLM (Large Language Model)
  - The brain of the system. Understands and generates human-like text.
- Function Calling
  - Lets LLMs trigger specific functions/tools to get accurate or real-time answers.
- Plugins/Tools
  - External utilities (like calculators, search engines, APIs) that expand the LLM's capabilities.
- RAG (Retrieval-Augmented Generation)
  - Pulls relevant documents or data before generating an answer—boosting accuracy.
- Agents
  - Autonomous LLM-driven entities that plan, decide, and collaborate to complete tasks.

What is an Agent in AI

- Definition:
  - An AI Agent is more than a chatbot – Its's an autonomous system that combines LLMs, tools, memory, and reasoning to complete tasks or reach goal.
- Key Traits:
  - Autonomous: Acts without constant human input
  - Perceptive: Reads environment or inputs
  - Goal-Driven: Works toward specific objectives
  - Interactive: Can talk to users or other agents
  - Action-Oriented: Uses tools, APIs, or outputs to act
- Where are Agents Used?
  - Virtual Assistants
  - Game AI
  - Robotics
  - Autonomous Vehicles
  - Multi-Agent Systems



Agent Frameworks

Framework	Maintainer	Language(s)	Key Strengths	Primary Use Cases
LangChain	LangChain Inc.,	Python, JavaScript	Strong LLM integration, memory, tool use, external API support	Chatbots, RAG, autonomous agents
Autogen	Microsoft Research	Python, C# (Works with Semantic Kernel)	Multi-agent collaboration, tool calling, task orchestration	Task delegation, cooperative agent workflows
CrewAI	CrewAI	Python	Role-based delegation, agent teamwork	Simulated teams, role-specific agents
MetaGPT	DeepWisdom	Python	Software team simulation (PM, Dev, Reviewer), modular pipeline	Software generation, project planning
Semantic Kernel	Microsoft	C#, Python, Java	Enterprise integration, plugin system, planner & memory support	Enterprise copilots, app-embedded LLM agents

Ways to Build Agents in Microsoft Ecosystem

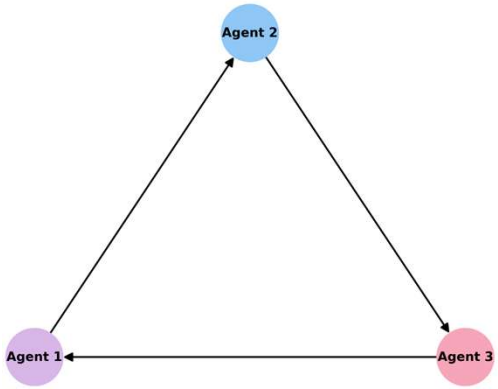
- Copilot Studio
  - No-code/low-code approach to building conversation agents
  - Integrated with Microsoft Teams, Power Platform, and Dynamics
- Azure AI Foundry
  - Enterprise-grade environment for building, evaluating, and deploying AI agents
  - Rich prompt flow, grounding, tool-use, and evaluation capabilities
- AutoGen Framework (Python/C#)
  - Multi-agent orchestration with human-in the-loop and tool calling
  - Great for scalable, customizable LLM agent workflows
- Semantic Kernel (C#/Python/Java)
  - Plug-in system for integrating LLMs into apps
  - Supports memory, planners, skill composition, and tool chaining

What Can an Agent Framework do?

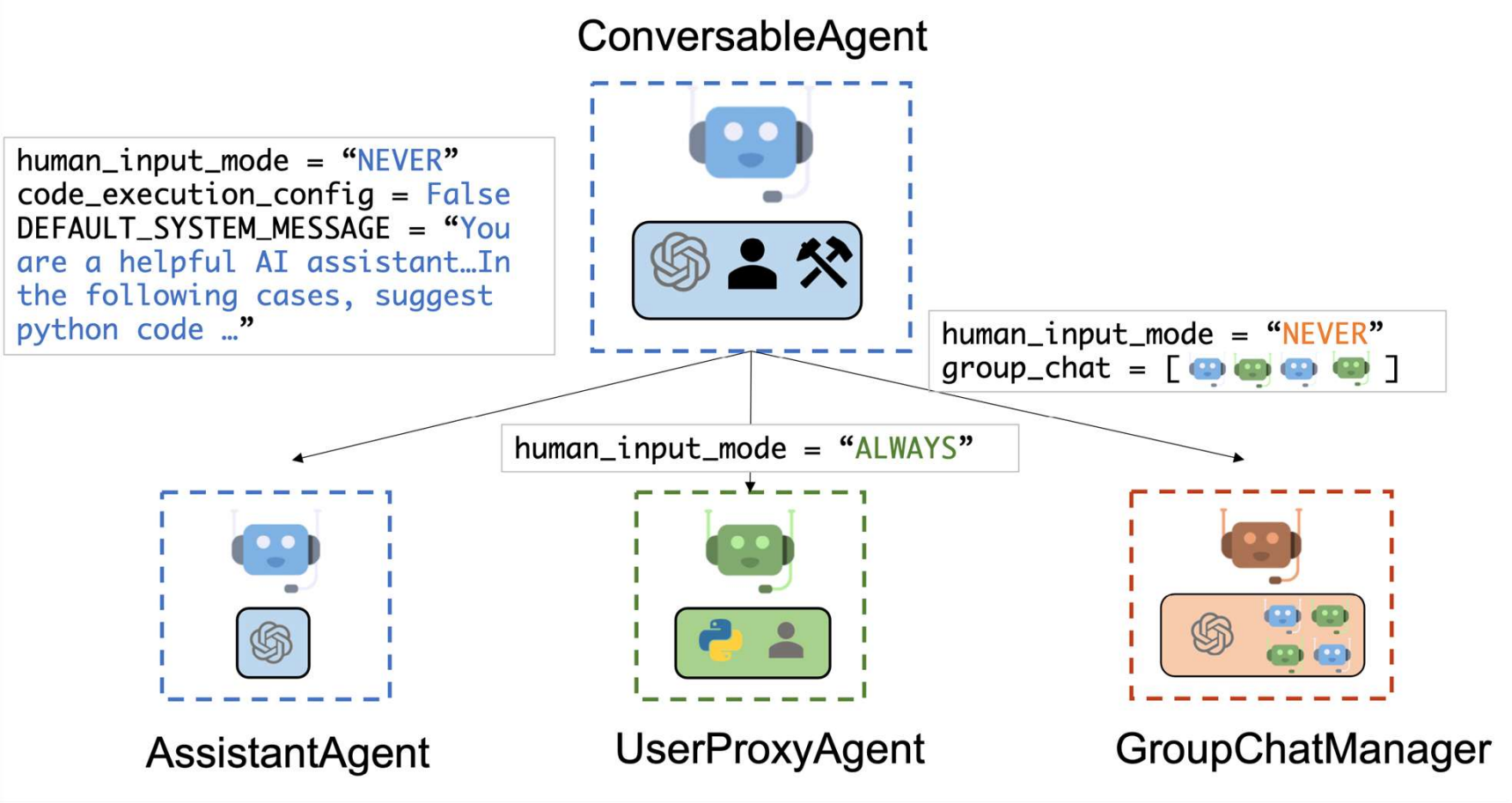
- Code, Execute, Supervise
  - Agents can write code, run it, and involve humans when needed.
- Customizable Agent Types
  - Mix and match LLMs, humans, and tools as agents.
- Conversable Interfaces
  - Agents can send and receive messages with a unified interface.
- Multi-Agent Collaboration
  - Supports group chats and coordinated tasks between agents.
- Sence, Decide, Act
  - Like autonomous cars: agents observe, plan and act to achieve goals.

Multi-Agent Collaboration

- Task Division
  - Task breakdown
- Role Assignment
  - Assign roles to each agent
- Agent Communication
  - Agents talk with each other (share information)
- Final Assembly
  - Combine all contributions from all agents

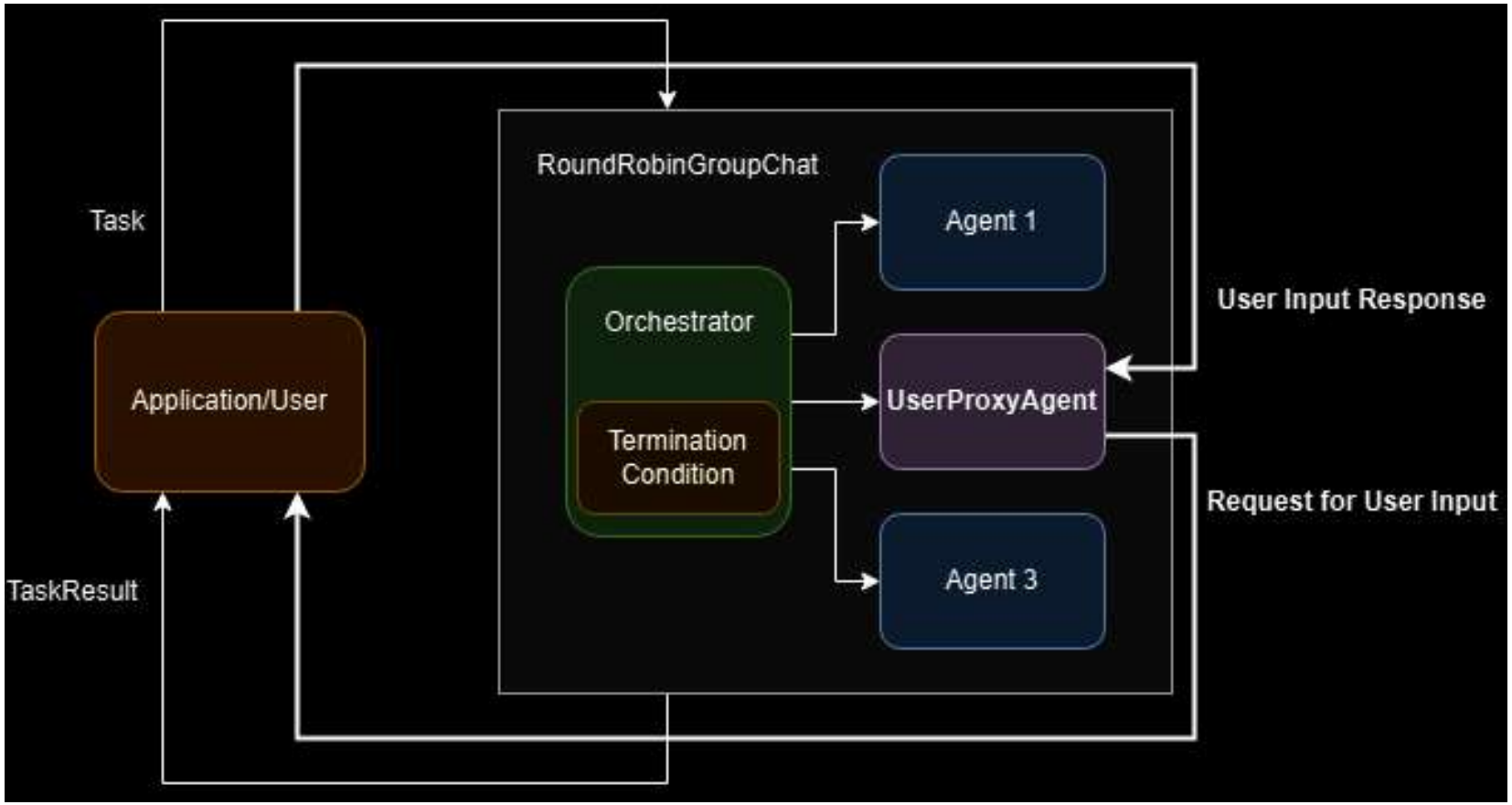


Built-in Agents in Autogen



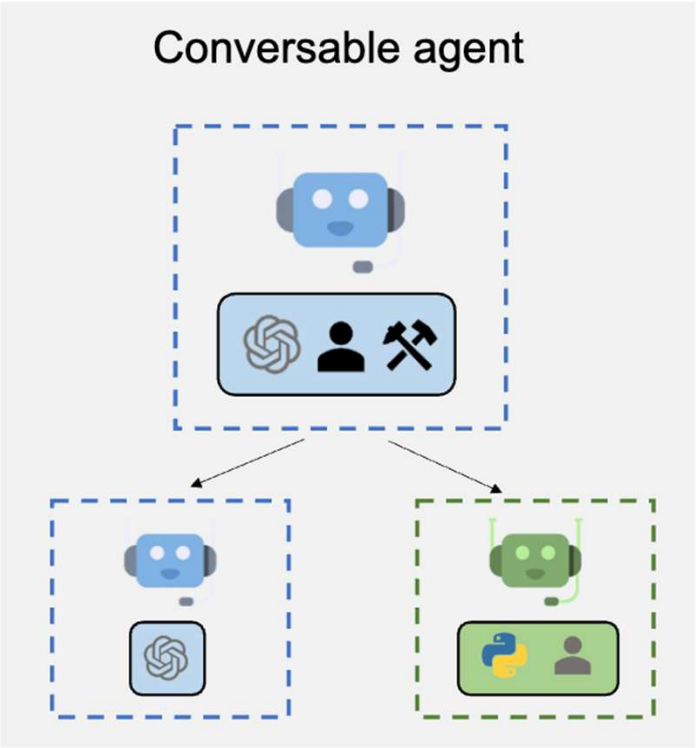
https://microsoft.github.io/autogen/0.2/docs/Use-Cases/agent_chat/

Human-in-the-loop in Autogen

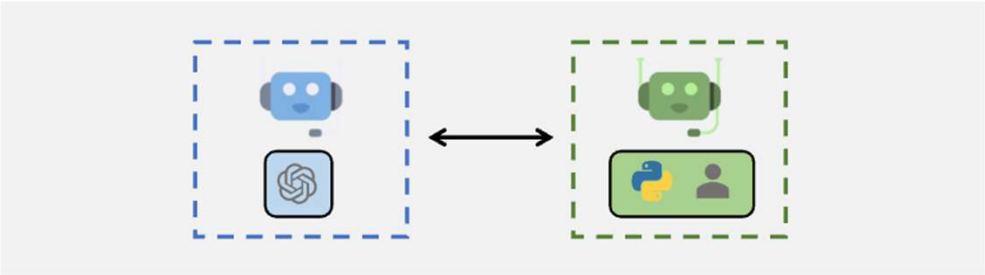


https://microsoft.github.io/autogen/stable/user-guide/agentchat-user-guide/tutorial/human-in-the-loop.html

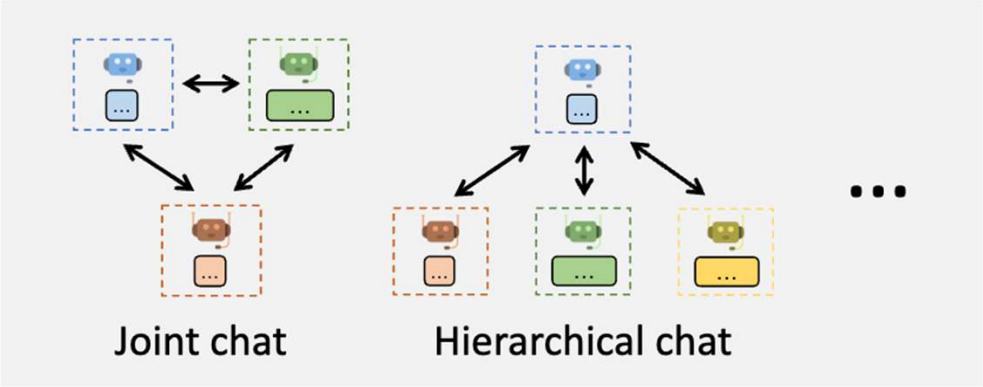
Multi-Agent Conversation



Agent Customization



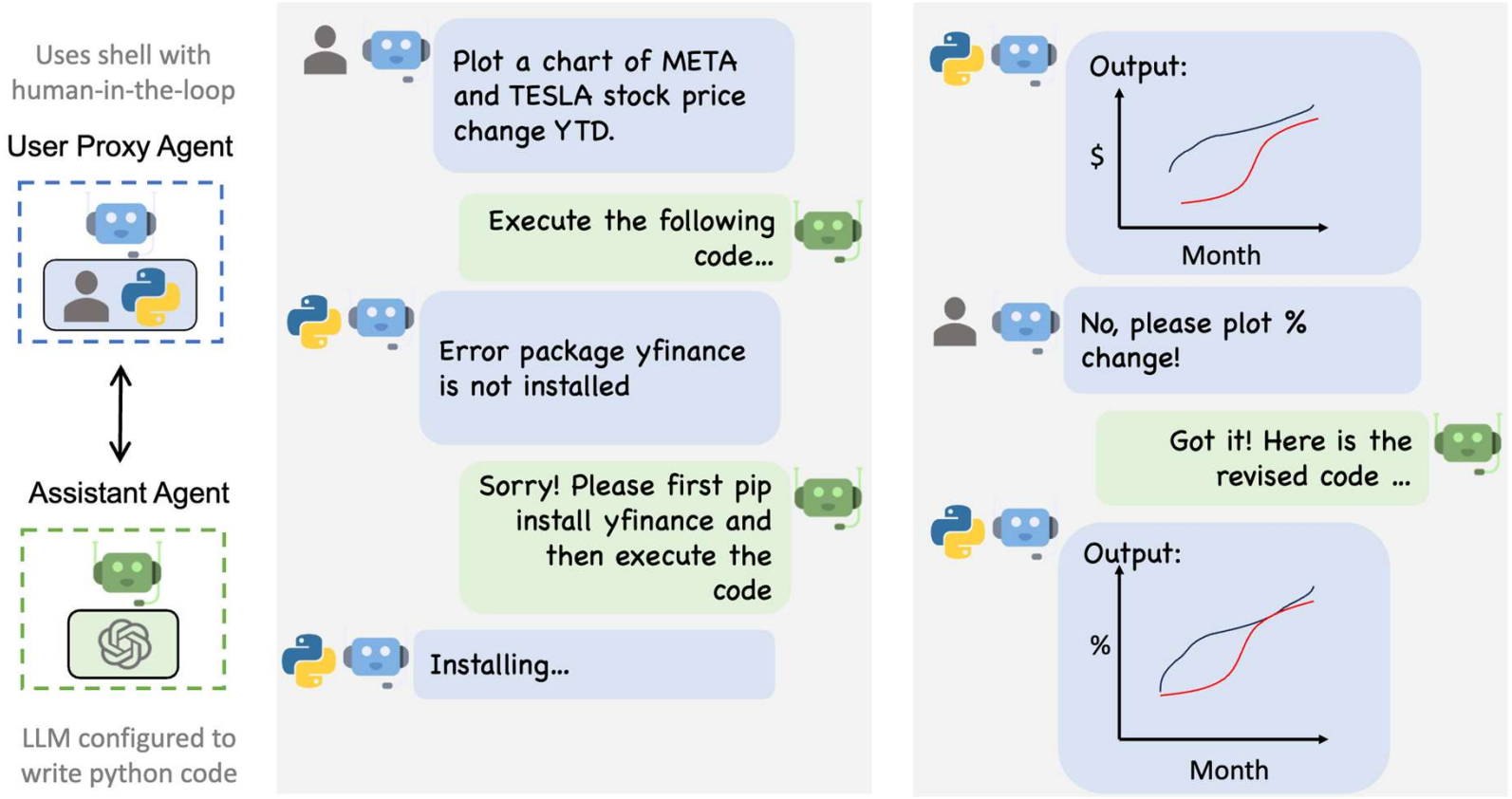
Multi-Agent Conversations



Flexible Conversation Patterns

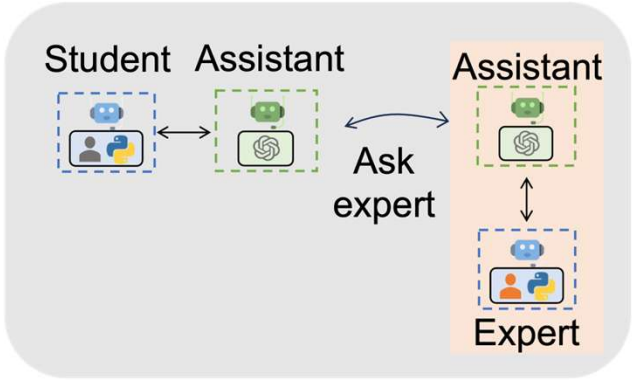
https://microsoft.github.io/autogen/0.2/assets/images/autogen_agentchat-250ca64b77b87e70d34766a080bf6ba8.png

Multi-Agent Conversation Flow

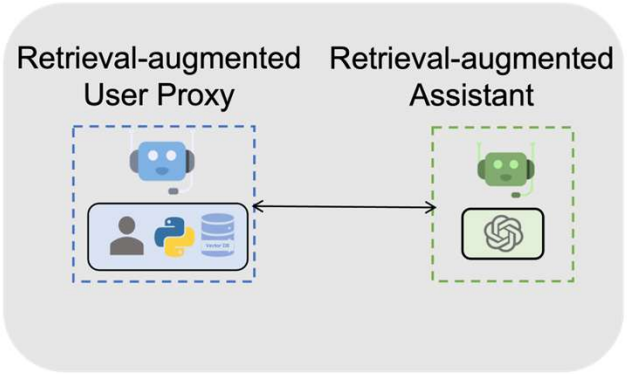


https://microsoft.github.io/autogen/0.2/assets/images/chat_example-da70a7420ebc817ef9826fa4b1e80951.png

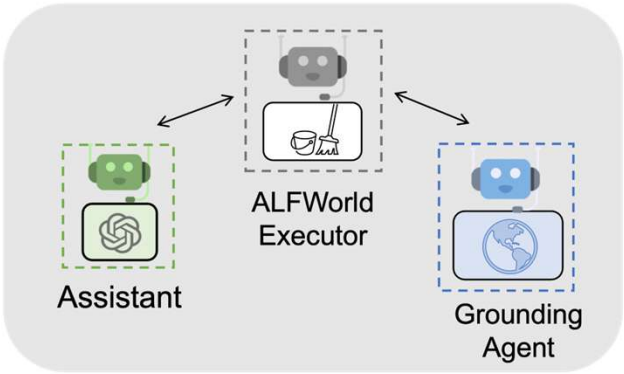
Multi-Agent Conversation: Diverse Application Implementation



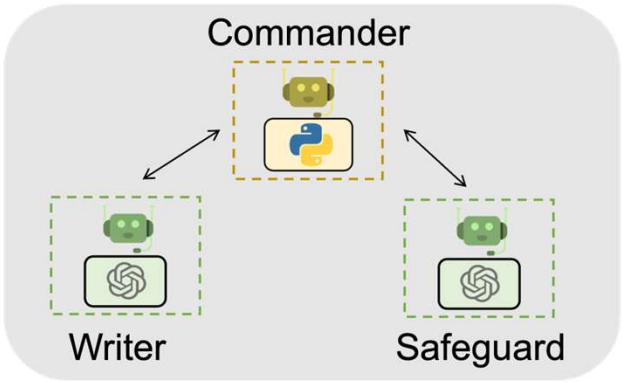
A1. Math Problem Solving



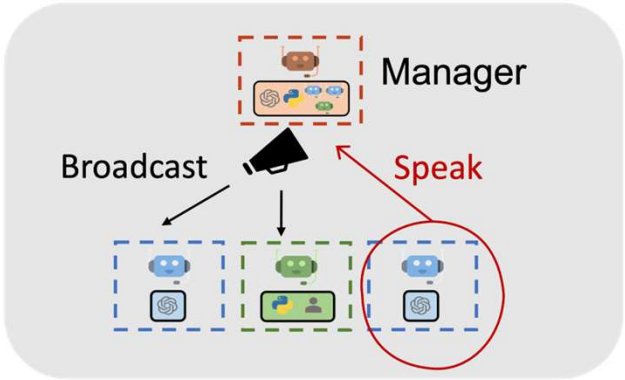
A2. Retrieval-augmented Chat



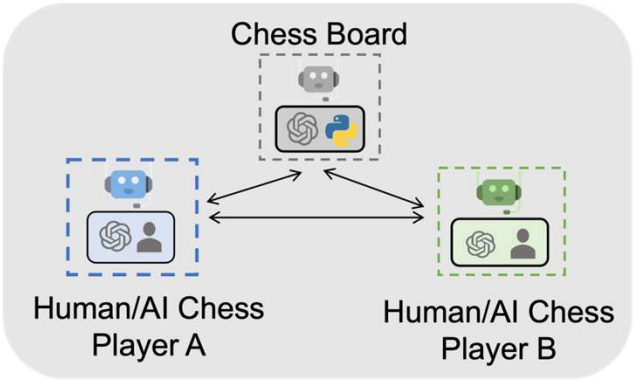
A3. Decision Making



A4. Multi-agent Coding



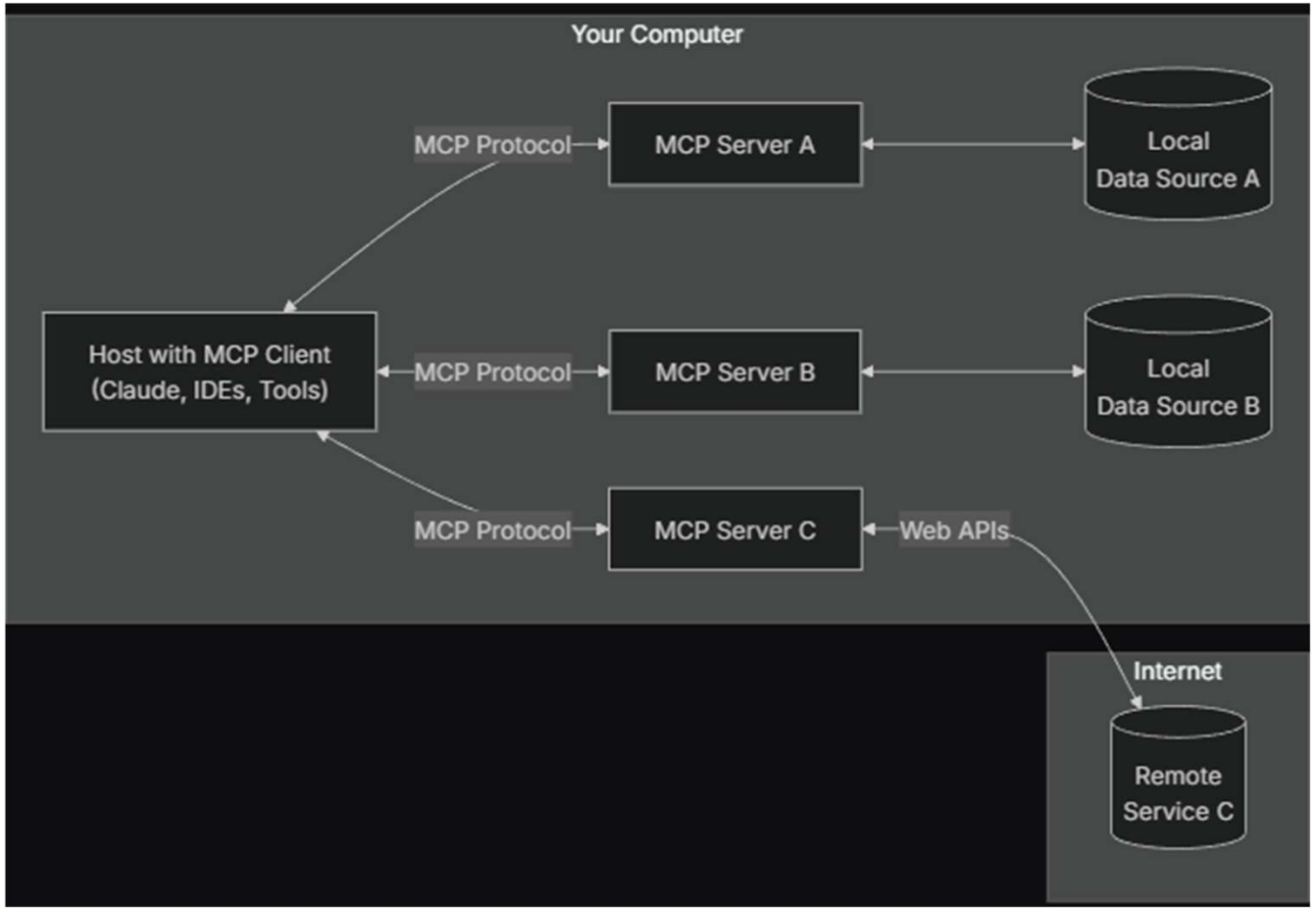
A5. Dynamic Group Chat



A6. Conversational Chess

https://microsoft.github.io/autogen/0.2/docs/Use-Cases/agent_chat/

Model Context Protocol (MCP)



https://modelcontextprotocol.io/introduction

1. Host with MCP Client (Claude, IDEs, Tools)
   - This is where the question originates.
   - A tool (e.g., CLI app, chatbot, or IDE plugin) with an MCP Client sends the weather request.

2. MCP Protocol → MCP Server C
   - The Host routes the request to MCP Server C using the MCP Protocol.
   - Why Server C? Because this agent is configured to access Web APIs (external services) – perfect for weather.

3. MCP Server C → Remote Service C (Internet)
   - MCP Server C parses the request (“Boston” as the location) and makes an external web API call to a weather provider (e.g., OpenWeatherMap).

4. Remote Service C → MCP Server C
   - The weather service returns data like:
     "Boston, 54°F, Partly Cloudy"

5. MCP Server C → Host with MCP Client
   - The result is sent back via the MCP Protocol.

6. Host Displays Result to User
   - The original tool (IDE, CLI, chatbot) shows:
     “The weather in Boston is 54°F and partly cloudy.”

Demo

References

- https://microsoft.github.io/autogen/stable/
- https://microsoft.github.io/autogen/0.2/docs/autogen-studio/getting-started/
- https://github.com/microsoft/autogen
- https://github.com/microsoft/ai-agents-for-beginners
- https://microsoft.github.io/autogen/dotnet/dev/core/index.html
- Copilot Agents: https://github.com/microsoft/agents-for-enhanced-customer-care-solution-accelerator
- Agentic Framework: https://github.com/microsoft/Multi-Agent-Custom-Automation-Engine-Solution-Accelerator
- Knowledge Mining: https://github.com/microsoft/Document-Knowledge-Mining-Solution-Accelerator

Thank you for your time and trust!

Boston Code Camp 38 – March 2025