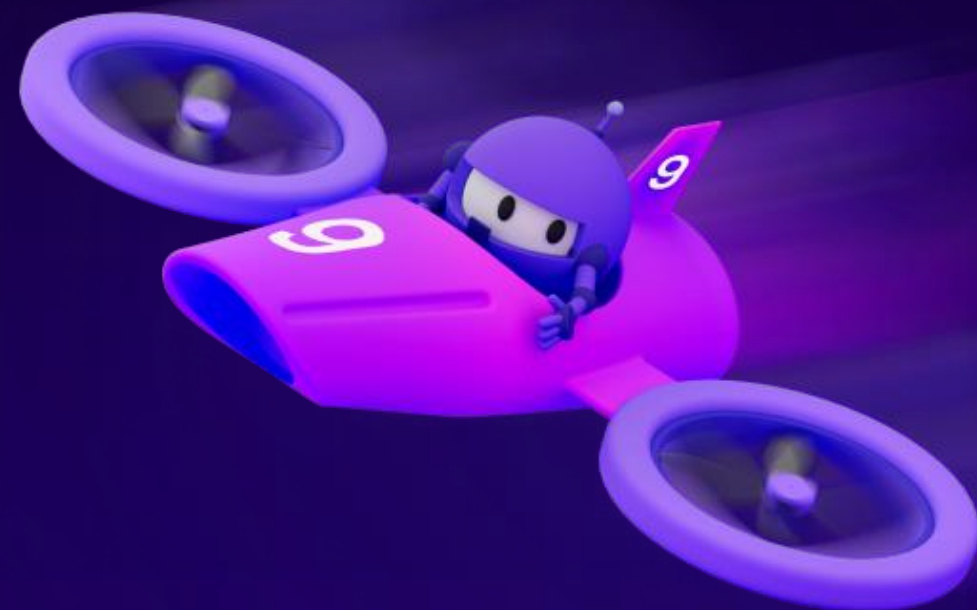


AutoGen

Build Agentic AI Apps with the Autogen framework

Phi Huynh

Technical Manager at NashTech
Microsoft MVP

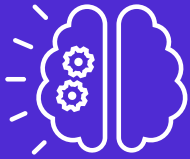


Agenda

- 1 What are Agentic AI applications
- 2 Introducing AutoGen framework
- 3 Understanding Conversation Patterns
- 4 Demo: Build Multi-agent Flows

This content is powered by Nitya Narasimhan, Eric Zhu from
Microsoft Research team.

Build Generative AI is challenging



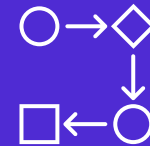
Large Language models

For rich content generation



Conversational Interactions

Driven by natural language processing capabilities

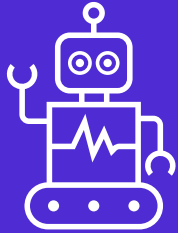


End-to-end workflows

From prompt-engineering to operationalization

Chat-based experiences require **human guidance (prompts)** and intervention to execute **complex tasks**, creating challenges in **workflow automation** and optimization

Agentic AI Application Development Can Help



Autonomous Agents

capable of planning & executing decisions



Task Execution Tools

identify and execute the right tools for each task



Conversational Workflows

coordinate actions across agents, user, environment

Agentic AI Applications use autonomous agents to execute **tasks on behalf of users**, interacting with their environment or remote services as needed, and coordinating actions with other agents for efficiency.

GenAI frameworks



LangChain



Semantic Kernel

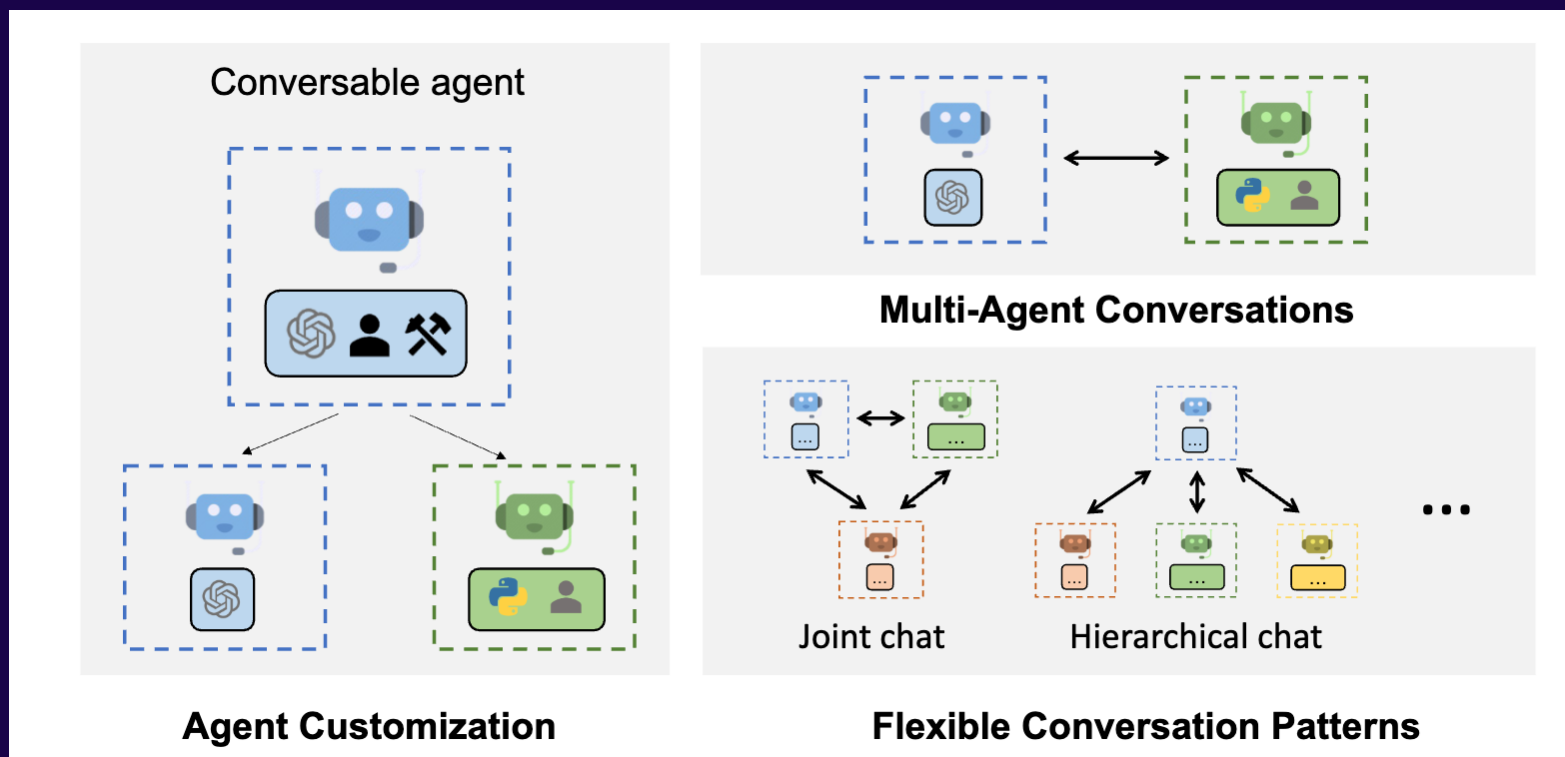


AutoGen



We will focus to AutoGen today

Introducing the AutoGen framework



Open-Source
Framework & Samples

Customizable
Conversable Agents,
LLMs

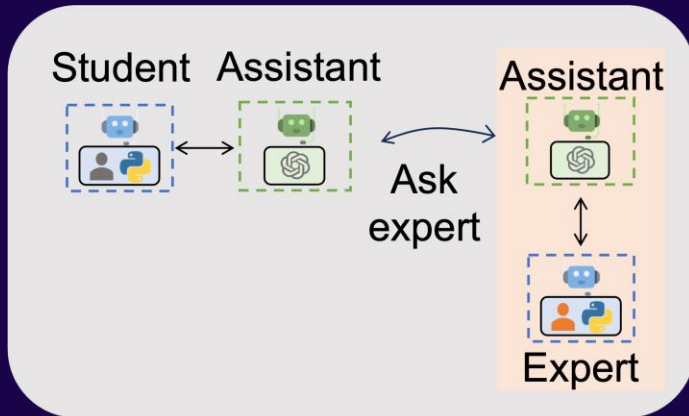
Research-Driven
Tools & Patterns

No-Code and Code-First
Development

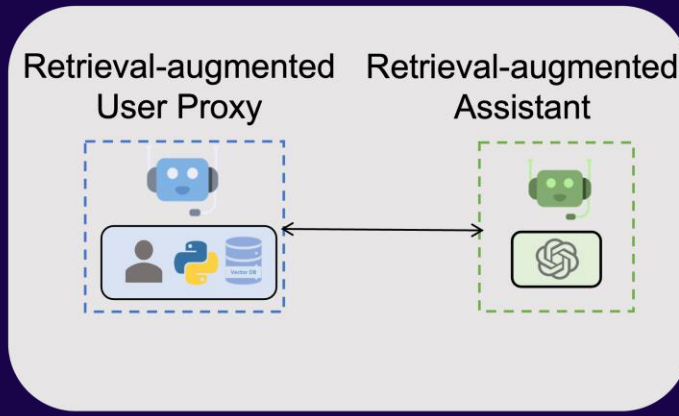
Docs: <https://aka.ms/autogen/website>

Discord: <https://aka.ms/autogen/discord>

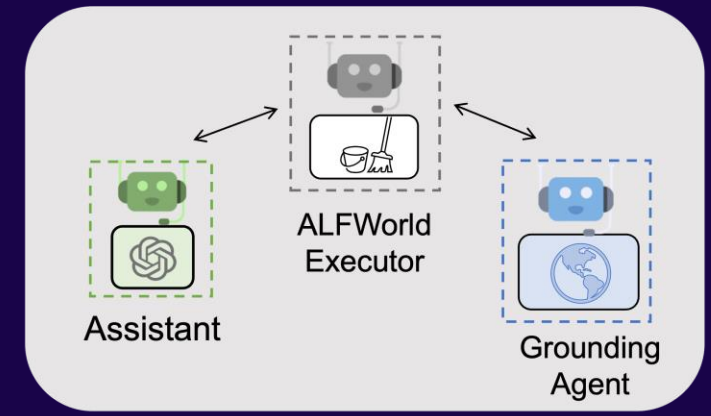
Build applications with AutoGen



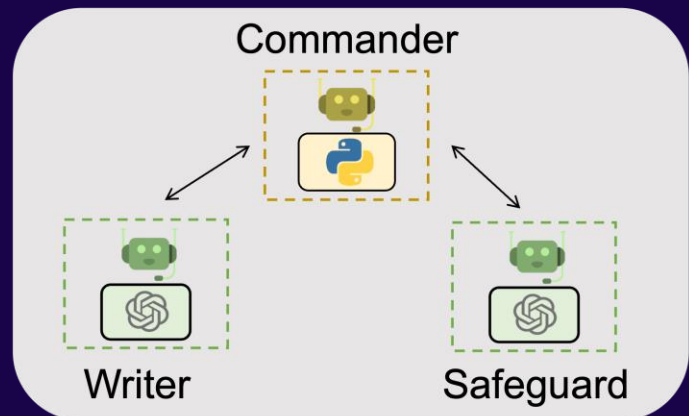
Math problem solving



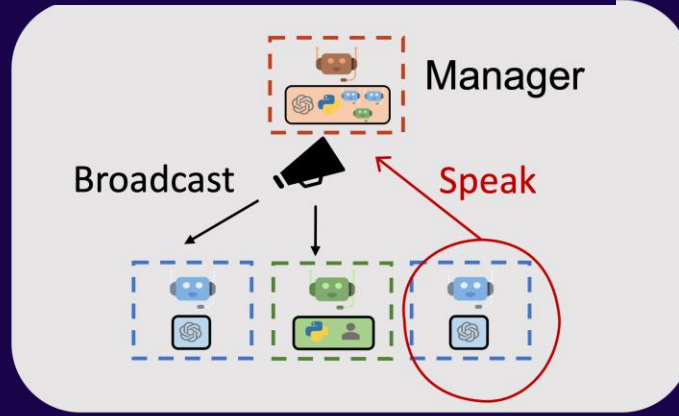
Retrieval augmented chat



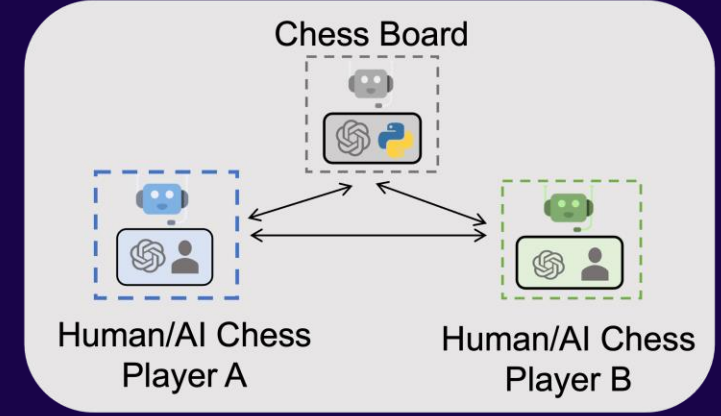
Decision making



Multi-agent coding



Dynamic group chat



Conversational chess

Supported Large-language models

- 1 OpenAI / Azure OpenAI
- 2 Ollama
- 3 Mistral
- 4 Gemini

Cool features

- 1 **Function calling**
- 2 **Middleware**
- 3 **Code executor (.NET, Python)**
- 4 **Multiple conversational patterns**

Get started with AutoGen

Agent setup – Code execution

```
var assistantAgent = new OpenAIChatAgent(  
    name: "assistant",  
    systemMessage: "You are an assistant that help user to do some  
tasks.",  
    chatClient: gpt-4o-mini  
    .RegisterMessageConnector()  
    .RegisterPrintMessage();
```

```
var userProxyAgent = new UserProxyAgent(  
    name: "user",  
    humanInputMode: HumanInputMode.ALWAYS)  
    .RegisterPrintMessage();
```

```
// start the conversation  
await userProxyAgent.InitiateChatAsync(  
    receiver: assistantAgent,  
    message: "Hey assistant, please do me a favor.",  
    maxRound: 100);
```

ConversableAgent

Agent (Protocol)

base agent

sends & receives messages

(conversations),

performs actions (and generates reply)

ConversableAgent (Agent)

generic agent can be configured as
user proxy

or as assistant. will send out reply
unless request is a termination
message.

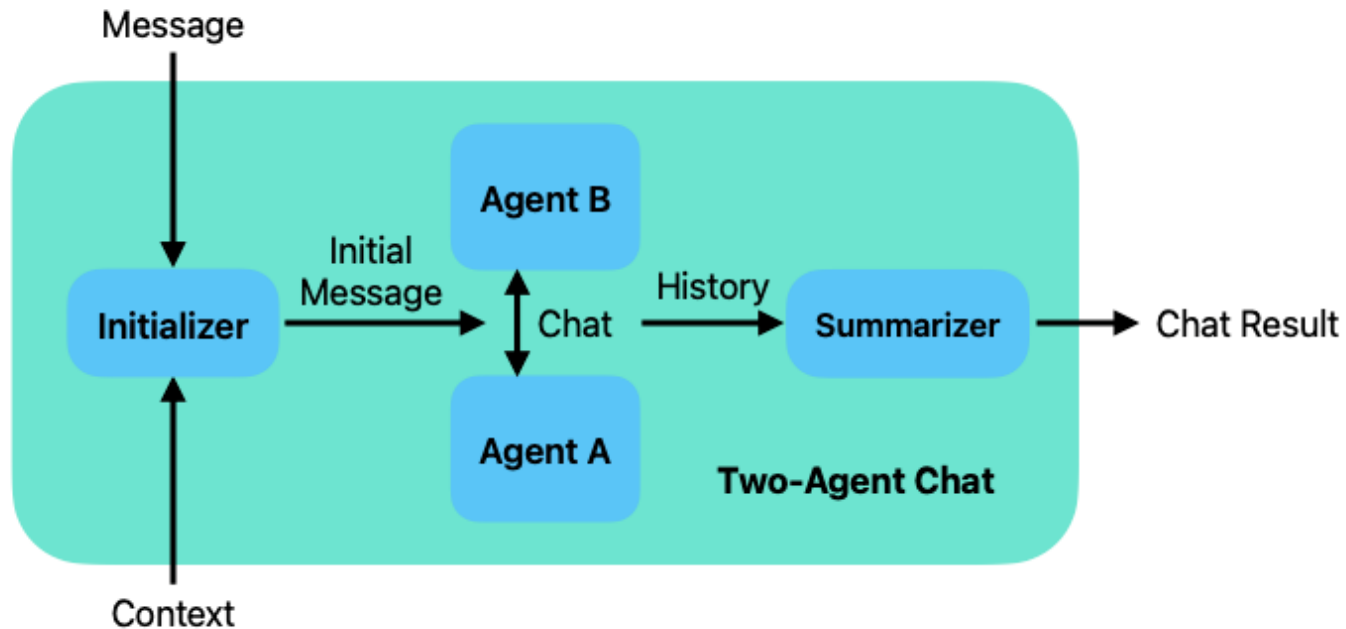
configure human-in-the-loop

intervention behavior (always, never,
termination only)

configure function_map (callable
functions) & **code_execution_config**
(local or docker)

Conversation patterns – 2 agents chat

E.g., "What is triangle inequality?"



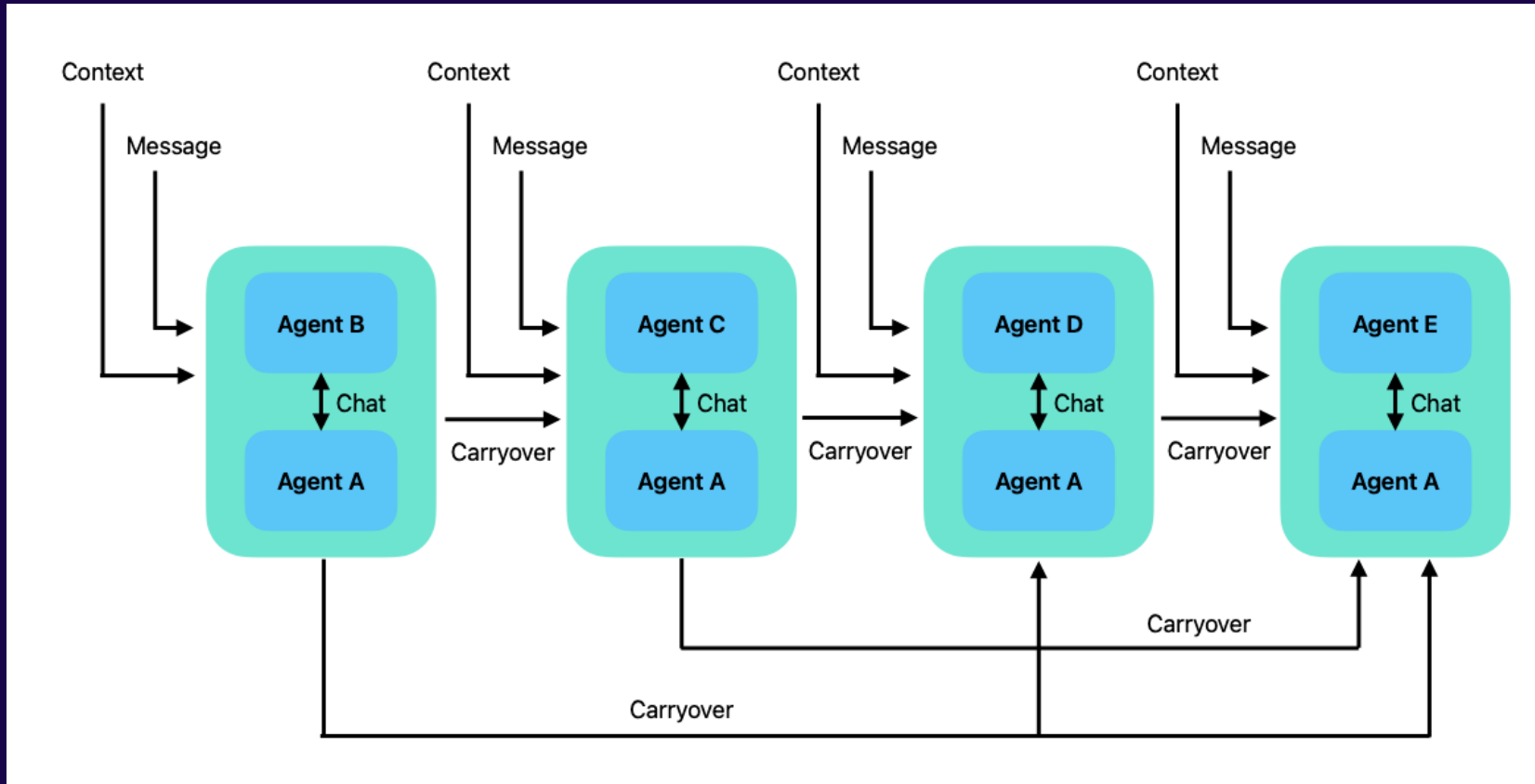
E.g., summary_method: reflection_with_llm

Two-agent Chat

Two agents share the same thread

- Each agent has a specific skill
- 2 agents communicate each other

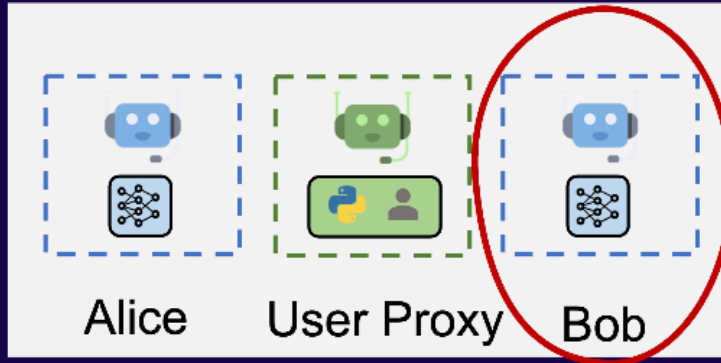
Conversation patterns – sequential chats



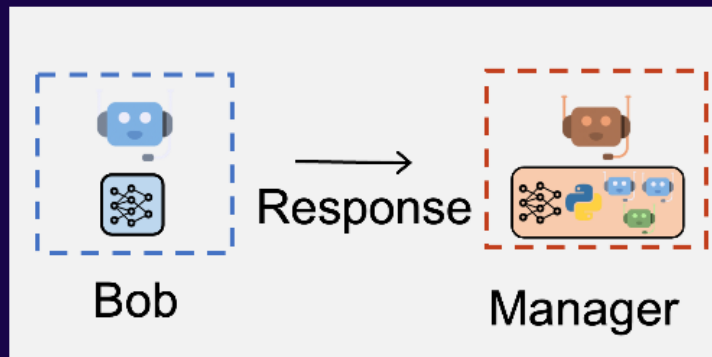
Sequential Chats

- Appropriate for workflows
- Agent A has ask agent to do the task
- Then Agent B carry over to Agent C to do the task
- And so on ...

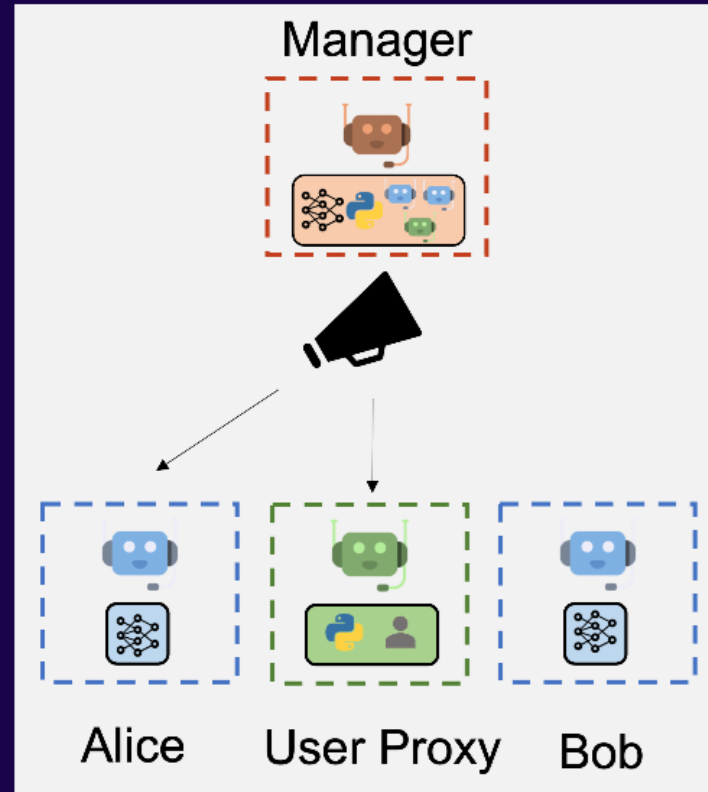
Conversation patterns – Group chat



1. Select a speaker



2. Ask speaker to response



3. Broadcast

Group chat

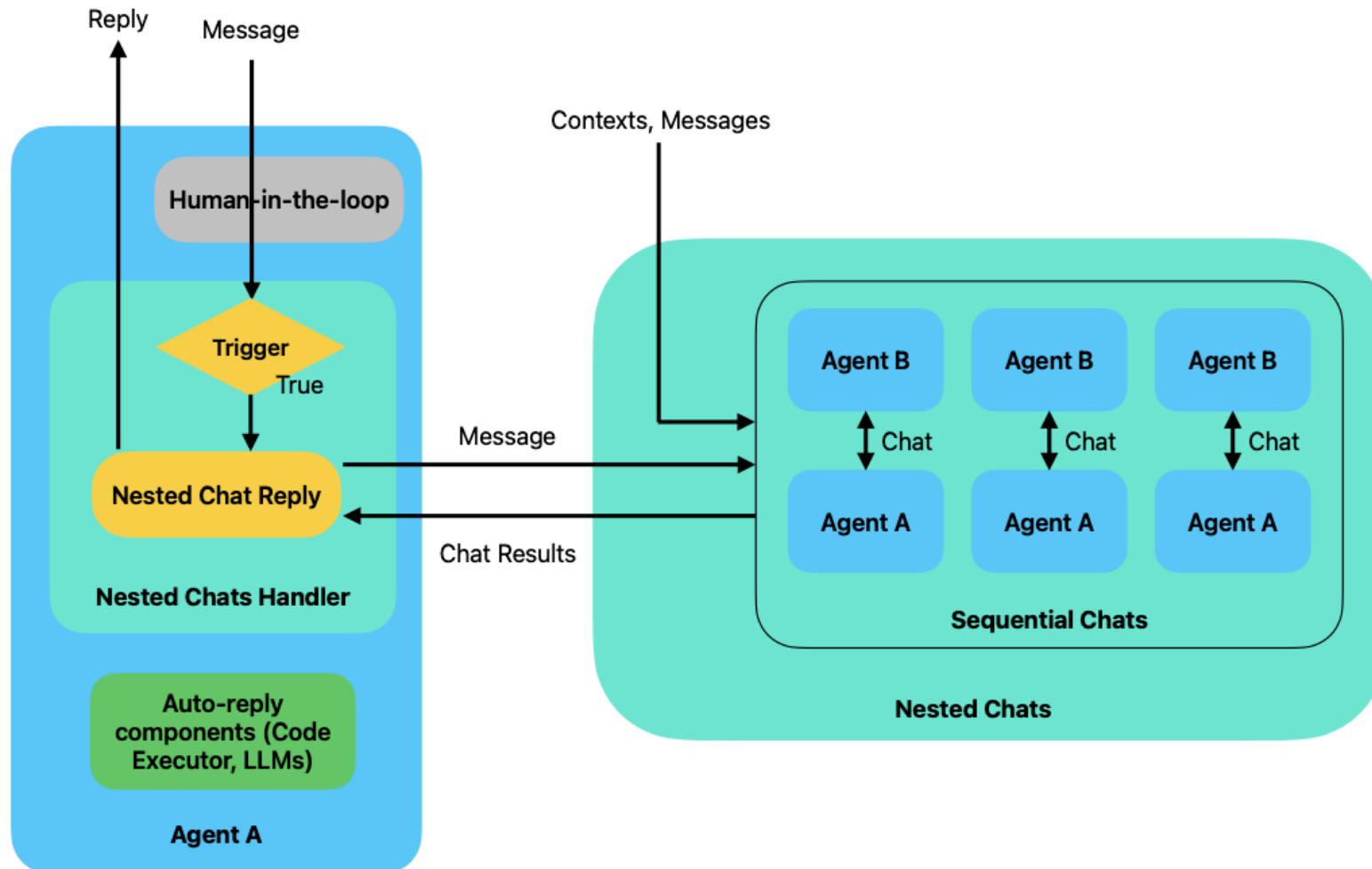
Agents participate in a single conversation thread.

- Speaker selected by a group chat manager
- Speaker responses
- Admin to collect human feedback

Conversation patterns – even more composable

Nested chat

Nested chat for complex scenarios.



AutoGen Studio

AutoGen Studio

The screenshot displays the AutoGen Studio interface. At the top, there's a navigation bar with 'Build Multi-Agent Apps', 'Build', 'Playground', and 'Gallery'. The 'Playground' tab is active. On the left sidebar, there's a 'Workflow' section with a 'General Agent Workflow' dropdown and a 'Sessions' section listing recent sessions. The main area shows a 'USER' prompt: 'create a 4 page pdf brochure on coffee from different parts of the world with some description of origins. E.g Ethiopian coffee may be in a glass on a table with a lush green forest in the background.' Below this, the 'AGENT' response states: 'The PDF brochure titled "Coffee_Brochure.pdf" has been successfully created. It includes images and descriptions of coffee from Ethiopia, Colombia, Brazil, and Vietnam, assembled into a 4-page document. Your brochure on coffee from different parts of the world is now ready. TERMINATE'. The 'Agent Messages' section shows a duration of 2 mins 39 secs. The 'Results' section displays 8 files: 'Coffee_Brochure.pdf', three generated images (77d9370c-1fe1-4658-a79a-ab789, f7a649b1-ce4a-4d5f-91a3-0085b, a24eea2a-8d7c-4855-a1a8-953b), and three Python scripts (6d3319d6-8041-49e1-acb5-439f, generate_more_images.py, create_pdf_brochure.py, generate_images.py). At the bottom, there's a text input field and a 'Blank slate? Try one of the example prompts below' section with buttons for 'Stock Price', 'Sine Wave', 'Markdown', and 'Paint'.

AutoGen Studio features

Define Skills

Create reusable functions, tools

Define Models

Define & configure required LLMs

Define Agents

Configure LLM, skills, behaviors

Define Workflows

Create agents, multi-agent conversations

Create Sessions

Test and validate agent workflows

Publish Sessions

Share sessions to a gallery to revisit



Get .NET 9



Download .NET 9
aka.ms/get-dotnet-9

Thank you

