AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation

Qingyun Wu[†], **Gagan Bansal**[*], **Jieyu Zhang**[±], **Yiran Wu**[†], **Beibin Li**[*]

Erkang Zhu[*], **Li Jiang**[*], **Xiaoyun Zhang**[*], **Shaokun Zhang**[†], **Jiale Liu**[∓]

Ahmed Awadallah[*], **Ryen W. White**[*], **Doug Burger**[*], **Chi Wang**[*1]

[*]Microsoft Research, [†]Pennsylvania State University

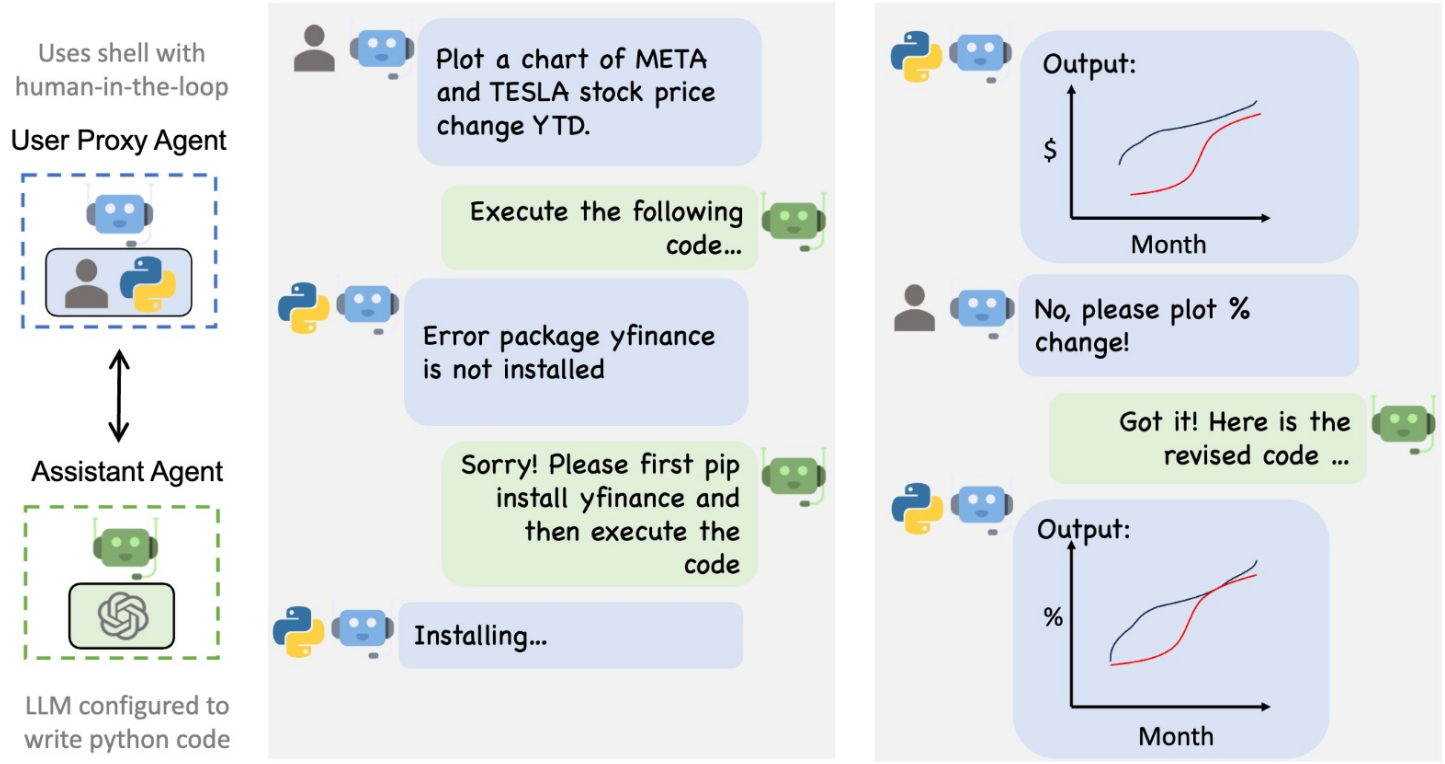
[±]University of Washington, [∓]Xidian University

Presented by:
John Tan Chong Min

Live Demo by:

Soham Sarkar (Crit)

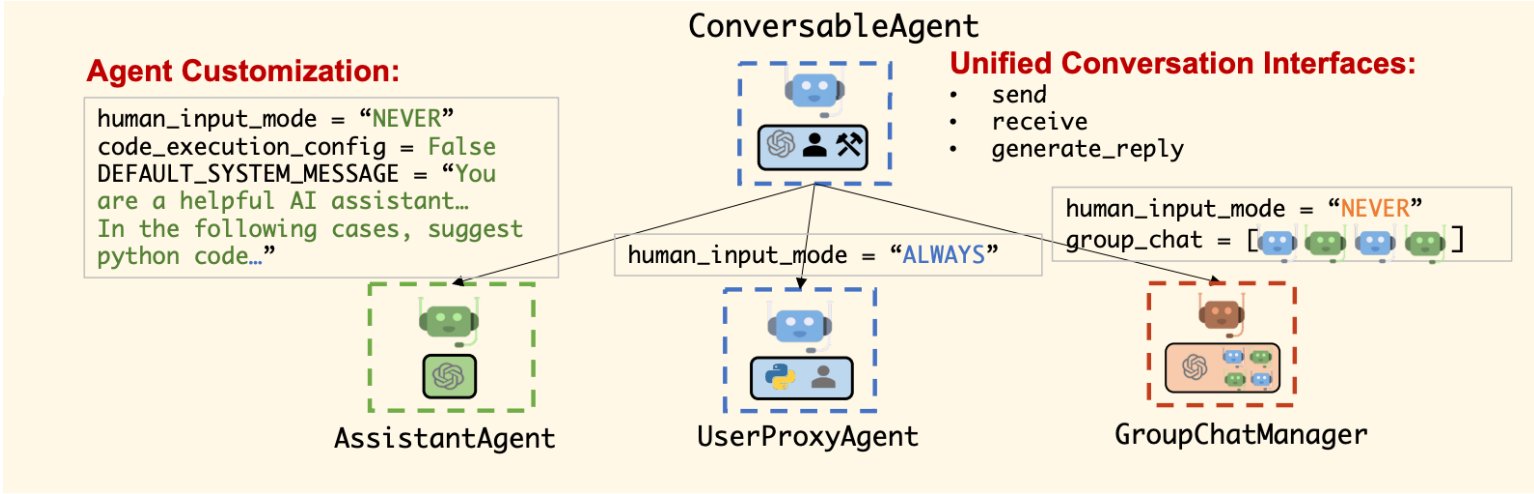
Imagine this



Agents

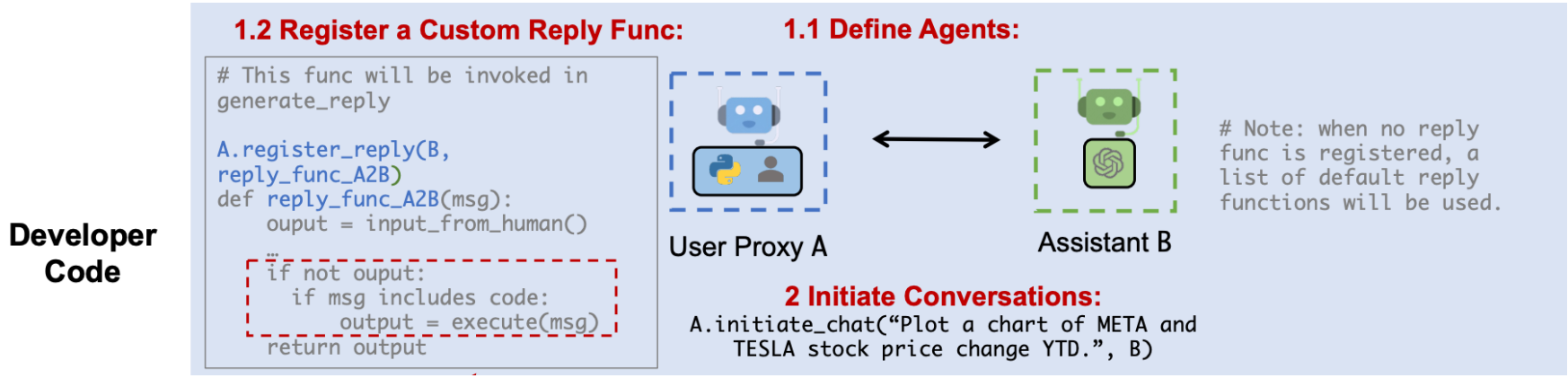
- Agent has generally 3 functions: **send, receive, generate_reply**
- You can also be an agent with human_input_mode = “ALWAYS”

AutoGen Agents



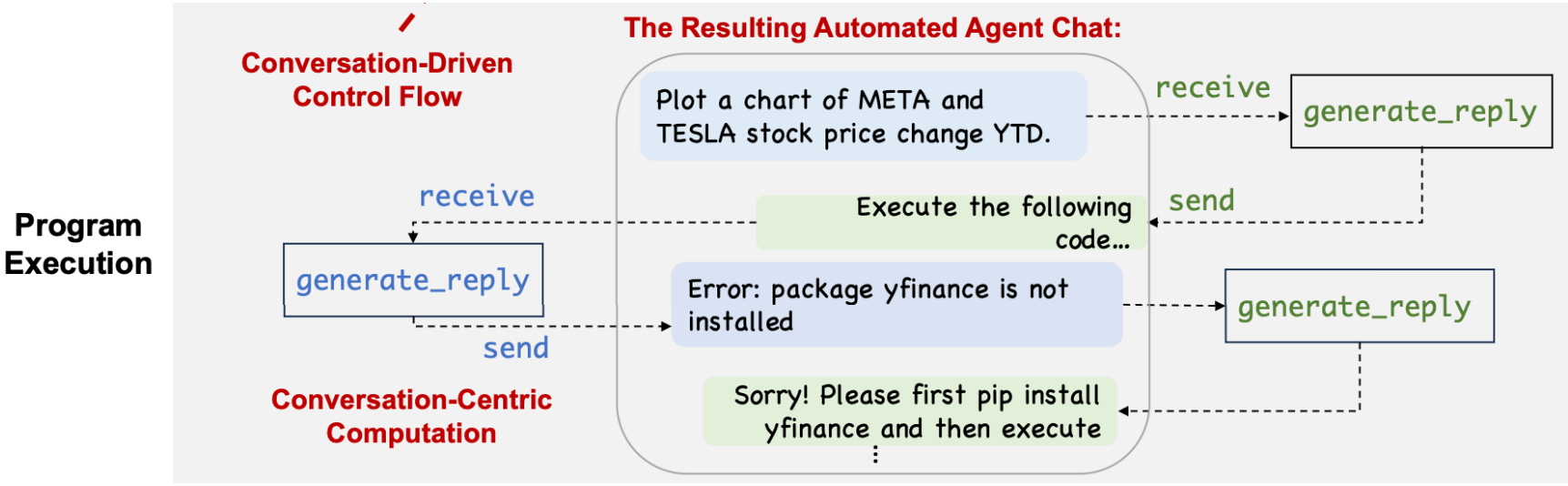
Flexible Customisable Functions

- Agent has template code which can be modified to suit your needs



Conversation-based Control Flow

- **receive:** Agent chooses what to receive
- **generate_reply:** Agent generates reply based on what it receives
- **send:** Agent sends reply to specified agents



Versatile Agent Structure

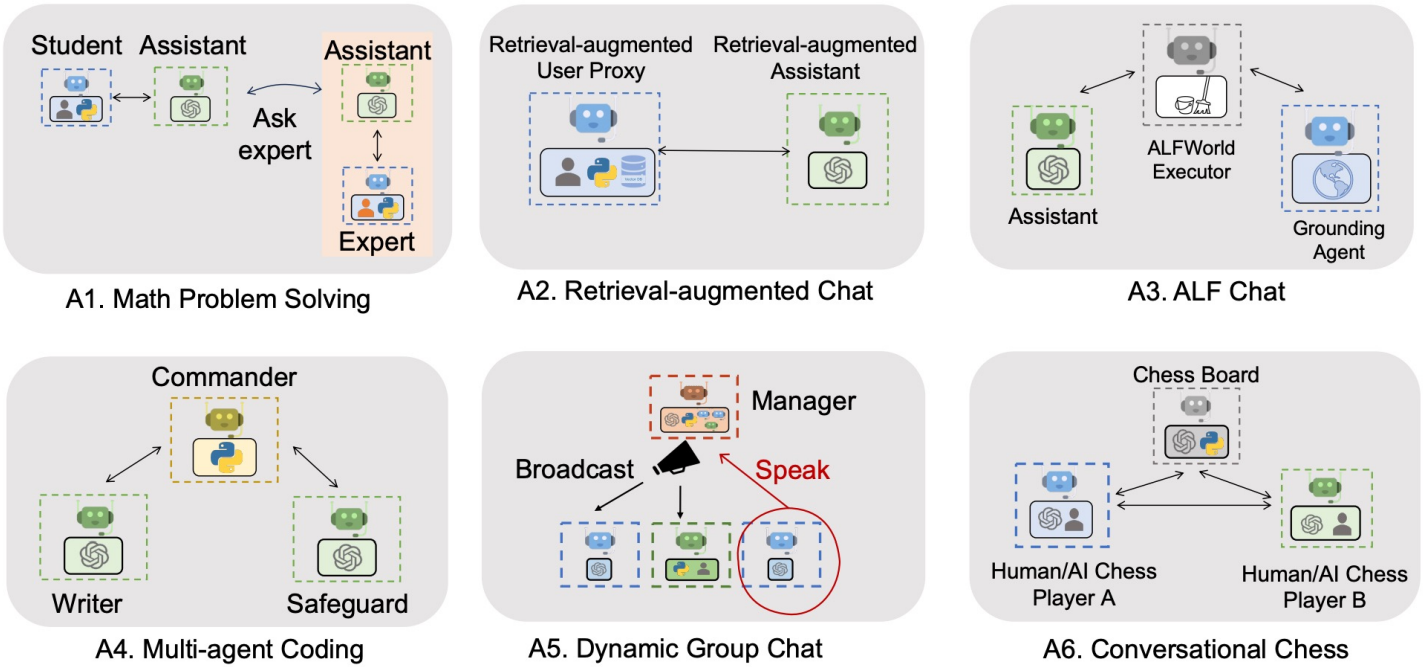


Figure 3: Six examples of diverse applications built using `AutoGen`. Their conversation patterns show `AutoGen`'s flexibility and power.

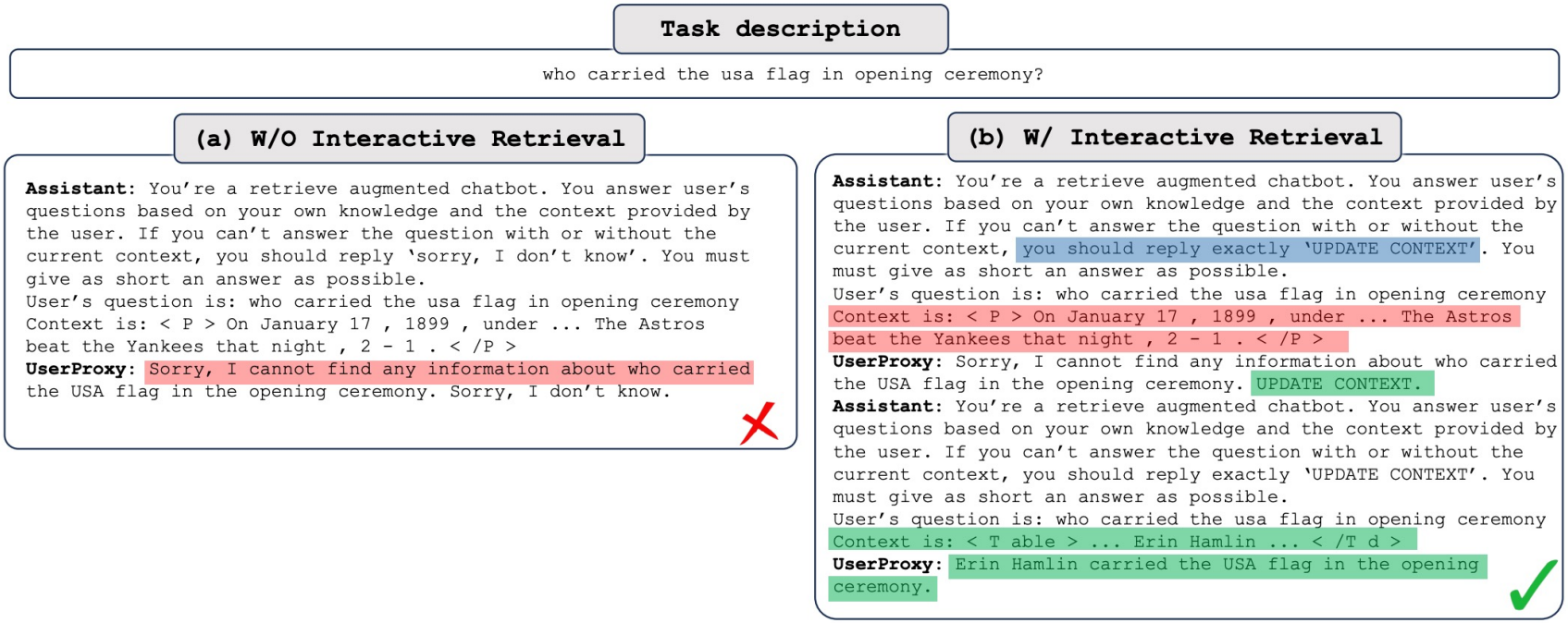
System Message

```
You are a helpful AI assistant. Solve tasks using your coding and language skills.

In the following cases, suggest python code (in a python coding block) or shell script (in a sh
coding block) for the user to execute.

1. When you need to collect info, use the code to output the info you need, for example, browse or
search the web, download/read a file, print the content of a webpage or a file, get the current
date/time. After sufficient info is printed and the task is ready to be solved based on your
language skill, you can solve the task by yourself.

2. When you need to perform some task with code, use the code to perform the task and output the
result. Finish the task smartly.

Solve the task step by step if you need to. If a plan is not provided, explain your plan first. Be
clear which step uses code, and which step uses your language skill.

When using code, you must indicate the script type in the code block. The user cannot provide any
other feedback or perform any other action beyond executing the code you suggest. The user can't
modify your code. So do not suggest incomplete code which requires users to modify. Don't use a
code block if it's not intended to be executed by the user.

If you want the user to save the code in a file before executing it, put # filename: <filename>
inside the code block as the first line. Don't include multiple code blocks in one response. Do not
ask users to copy and paste the result. Instead, use 'print' function for the output when relevant.
Check the execution result returned by the user.

If the result indicates there is an error, fix the error and output the code again. Suggest the
full code instead of partial code or code changes. If the error can't be fixed or if the task is
not solved even after the code is executed successfully, analyze the problem, revisit your
assumption, collect additional info you need, and think of a different approach to try.

When you find an answer, verify the answer carefully. Include verifiable evidence in your response
if possible.

Reply "TERMINATE" in the end when everything is done.
```

- Code as part of conversation enclosed in python coding block for regex extraction later
- Could perhaps be better formatted as JSON? (e.g. StrictJSON)

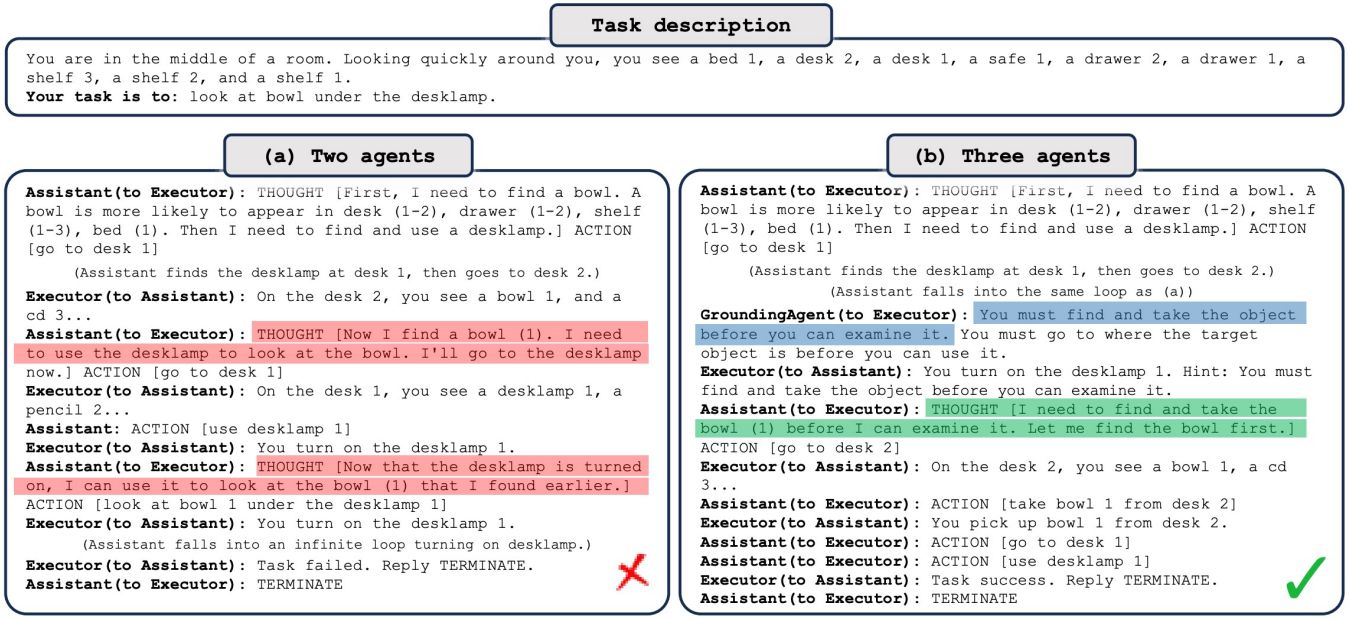
Retrieval Augmented Generation on demand via Agentic Interaction

- ”Update Context” is cue for RAG



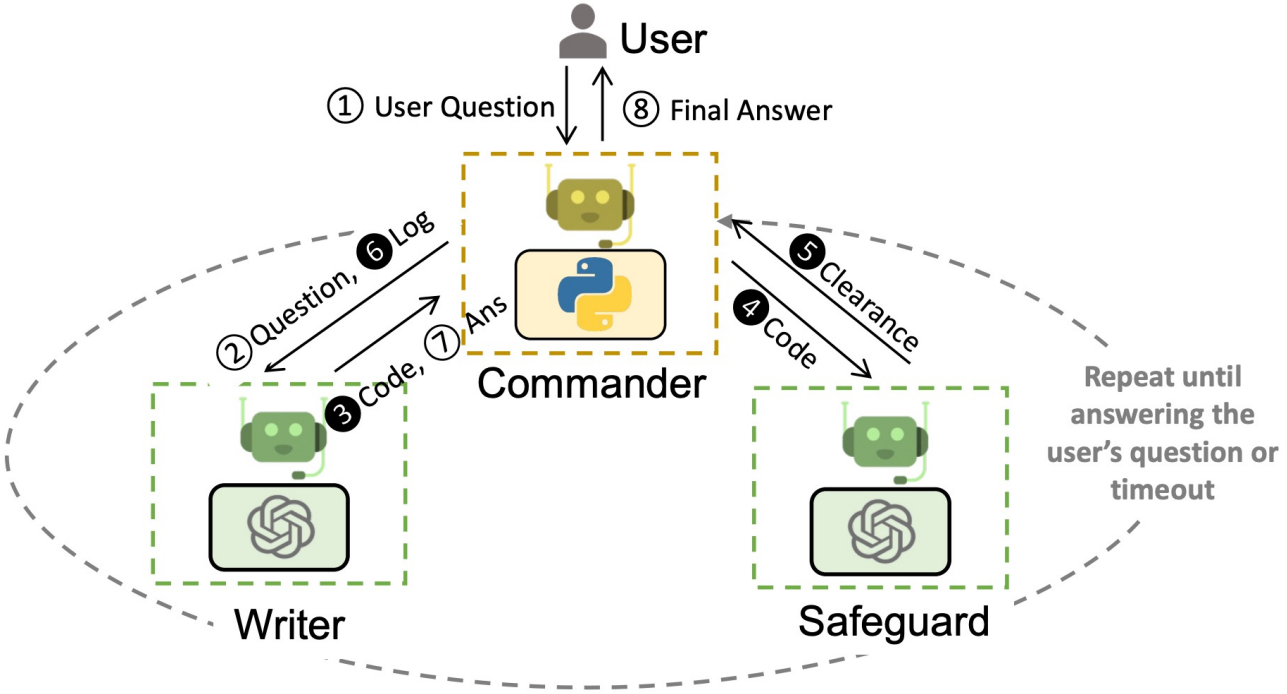
Grounding in Environment Reminder can be an agent

- What is typically a system message can also be an agent
  - Efficient?
- Conversation flow is akin to reflection and can improve performance



Red-teaming and Blue-teaming as a System of agents!

- One agent writes code, one agent safeguards against code
- Repeat until Commander is satisfied



Advanced Topics

Dynamic Flow!

Turn Order

- In a conversation, important to have only one person speaking at one time

- How do we decide whose turn to speak, and whether to skip the agents when not required?

Solution: Manager aka Flow Regulator

- Can either be turn-based (cycle through agents deterministically), or task-based (manager selects agent to respond based on current context)

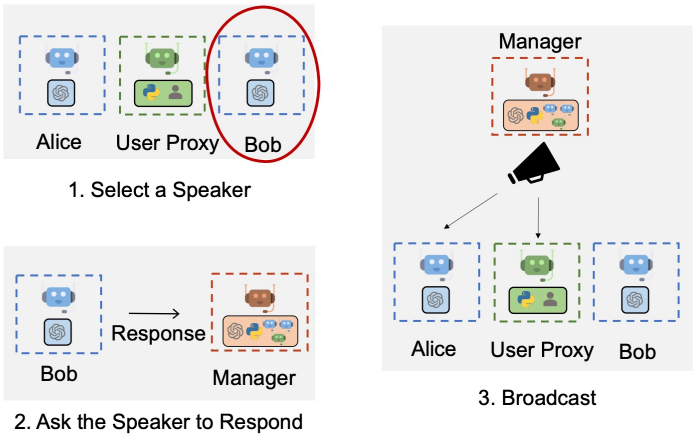


Figure 12: A5: Dynamic Group Chat: Overview of how `AutoGen` enables dynamic group chats to solve tasks. The Manager agent, which is an instance of the `GroupChatManager` class, performs the following three steps–select a single speaker (in this case Bob), ask the speaker to respond, and broadcast the selected speaker's message to all other agents

More Agents = Better?

- Generally, more agents can help improve performance (to a limit)
- **Task-based speaker selection performs poorer** – LLMs may not know how to select based on tasks well enough
- More agents = more context = more history
  - LLMs fail at long context

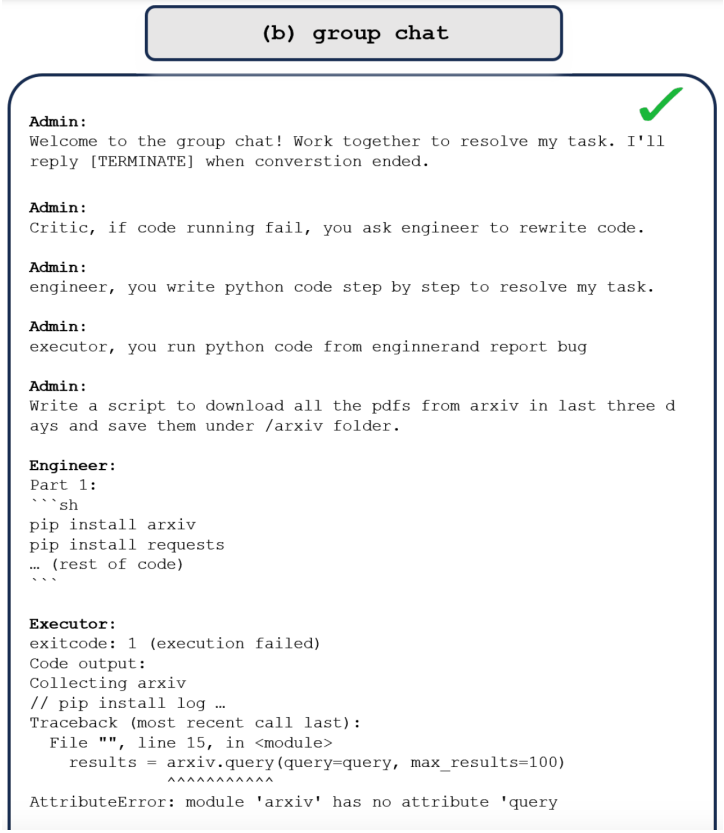
Table 5: Number of successes on the 12 tasks (higher the better).

Model	Two Agent	Group Chat	Group Chat with a task-based speaker selection policy
GPT-3.5-turbo	8	**9**	7
GPT-4	9	**11**	8

Table 6: Average # LLM calls and number of termination failures on the 12 tasks (lower the better).

Model	Two Agent	Group Chat	Group Chat with a task-based speaker selection policy
GPT-3.5-turbo	9.9, 9	5.3, 0	4, 0
GPT-4	6.8, 3	4.5, 0	4, 0

Self-correcting code with a suite of agents



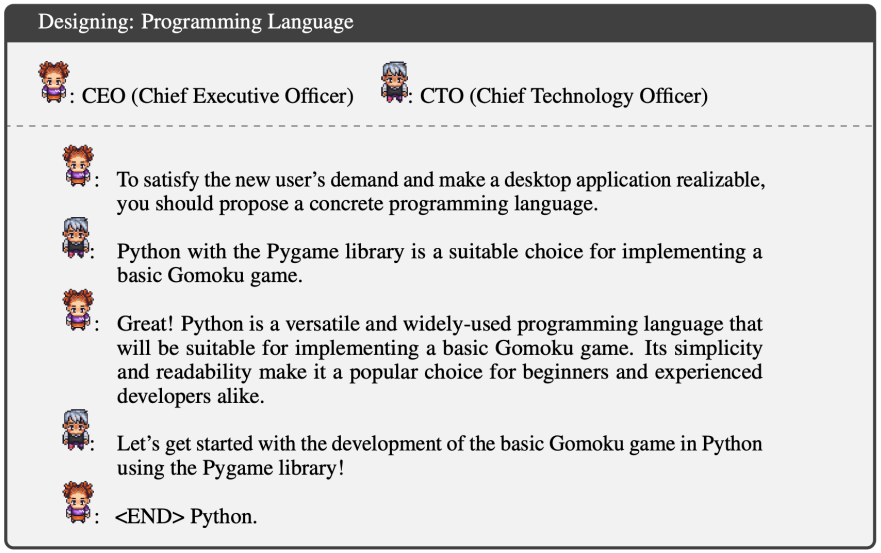
Pseudo Multi-Agent Conversation

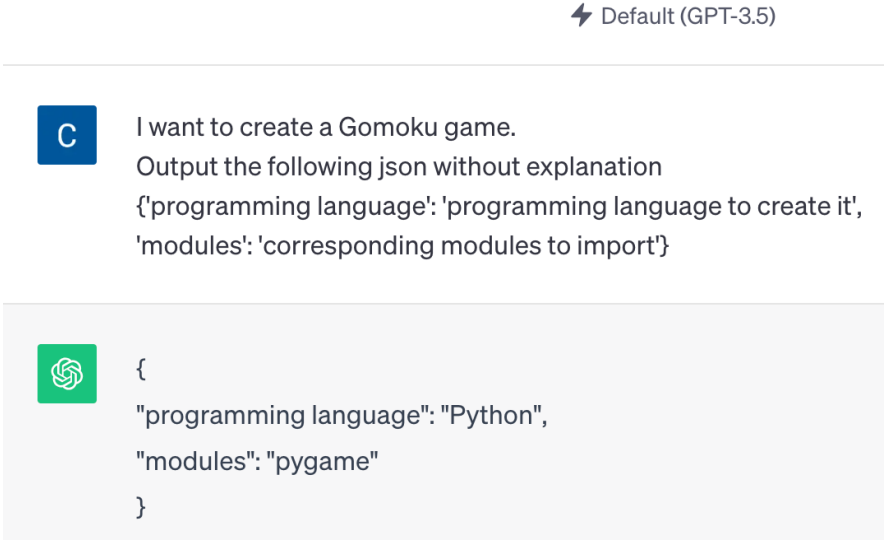
- In addition to conversation with predefined flow, AutoGen also supports **dynamic conversation flows with multiple agents** (indirectly – there is still a predefined flow to keep, but you can branch separate processes)

- AutoGen provides two general ways to achieve this:
  - **1) Customized generate reply function**: within the customized generate reply function, one agent can hold the current conversation while invoking conversations with other agents depending on the content of the current message and context.

  - **2) Function calls**: In this approach, LLM decides whether or not to call a particular function depending on the conversation status. By messaging additional agents in the called functions, the LLM can drive dynamic multi-agent conversation

Recap ChatDev Presentation: Is this really necessary?





ChatDev

Mine

My Critique on "**Communicative Agents for Software Development. Qian et al. 2023.**"

My Takeaways from AutoGen

- More agents may not be good – **context length should be short** for it to work too
- **Modular agents to do different tasks** is a good idea
- Conversation-style is **not necessary** if we already have structured approaches

Live Demo (by Crit)

Questions to Ponder

- Is the conversation interface the most efficient at getting things done?

- Should the number of agents be flexibly designed to be 2 or more depending on the situation, or just limited to 2 as per the CAMEL framework used in ChatDev?
  - Communicative Agents for “Mind” Exploration of Large Language Model Society (CAMEL) (Li et al, 2023)

- How to carry over learnings of one agent to another instance?