AutoGen

Enabling Next-Gen LLM Applications via Multi-Agent Conversations : [arXiv] [GitHub]
Qingyun Wu et al., COLM 2024

Shikhar Shiromani

Agentic AI for Oncology

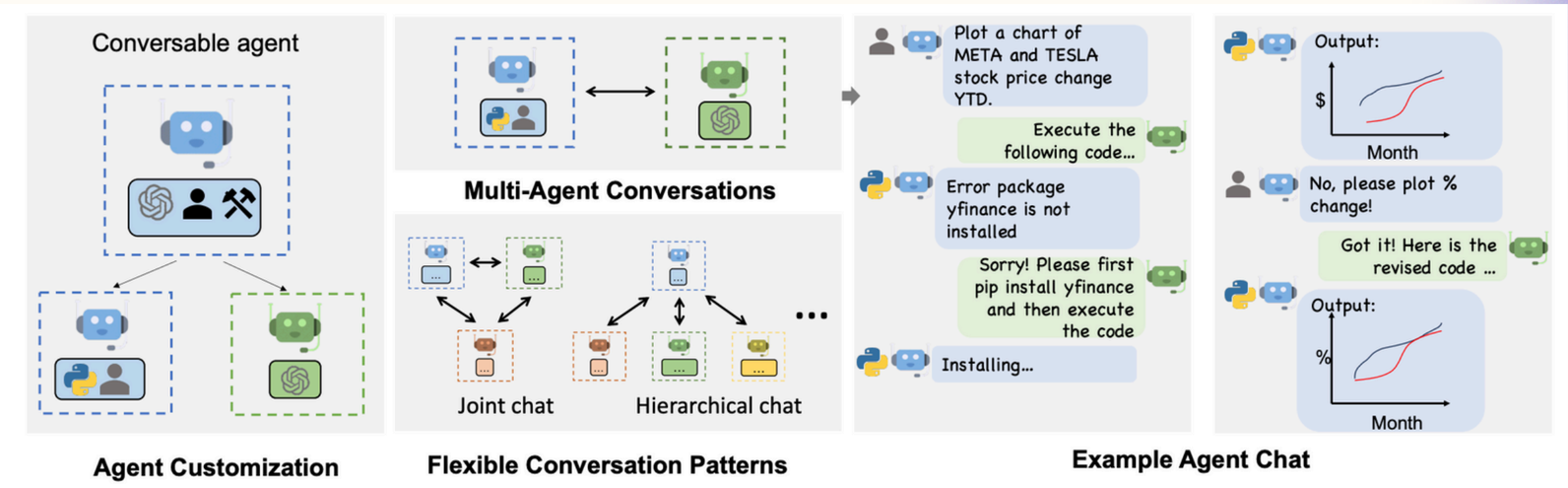
Why AutoGen?

- Single-agent LLMs hit limits in complex, multi-step tasks
- Many real problems need multiple specialized roles
- Existing multi-agent frameworks are rigid or code-heavy

Solution : *Multi-Agent Collaboration*

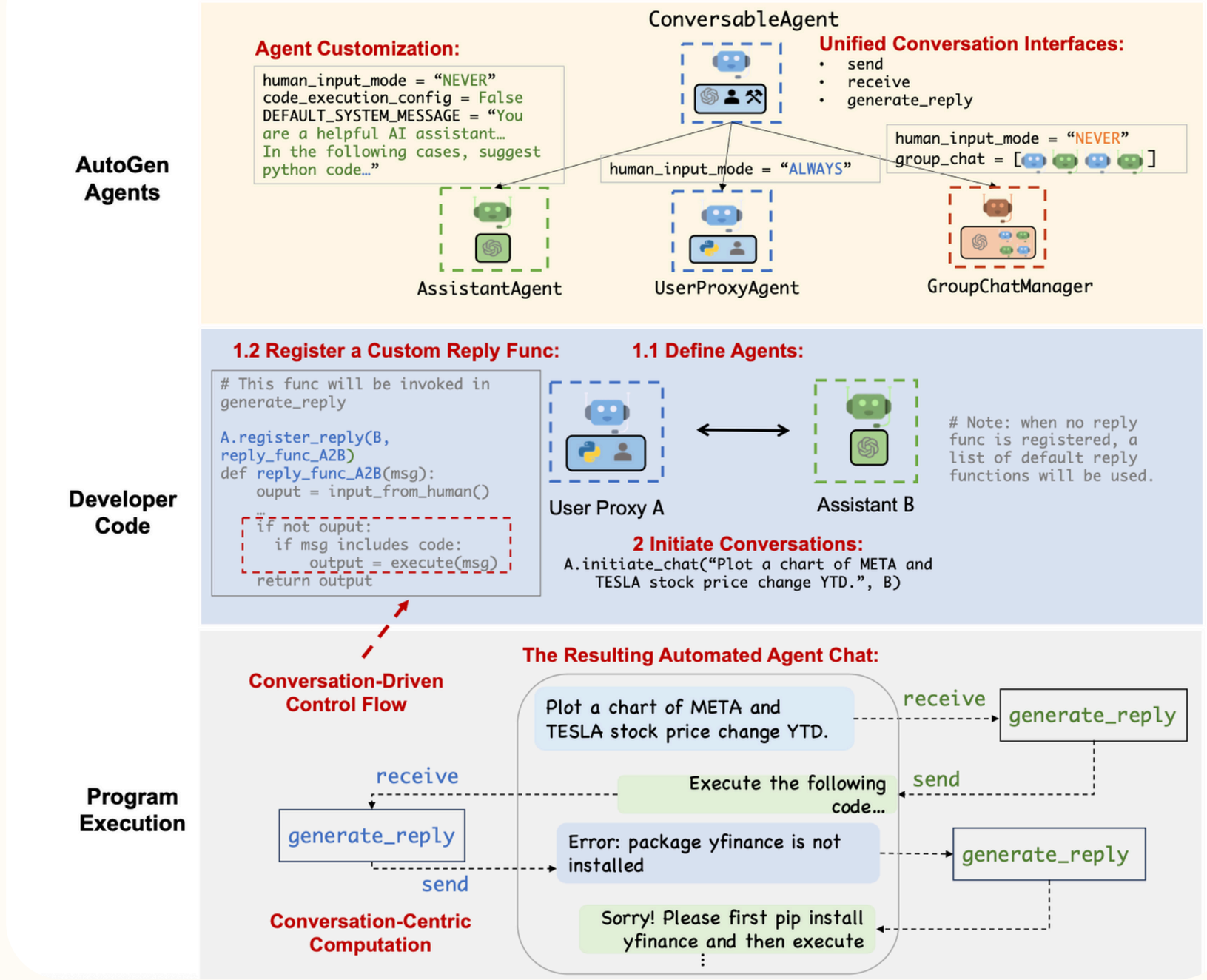
- Designing capable and customizable agents for effective multi-agent collaboration
- Creating a unified interface to support diverse conversation patterns

Core Idea = Multi-Agent Apps via Conversations



- Conversable Agents: LLM, human, or tool-driven
- Conversation Programming: define workflows via send/receive
- Modular, human-in-the-loop friendly

The Framework



Key Applications

- Math Problem Solving => Solver + Checker loop → better MATH benchmark scores
- Retrieval-Augmented QA => Iterative retrieval → higher factual accuracy
- ALFWorld Decision-Making => Grounding agent → +15% success rate
- Supply Chain Optimization => 3-agent workflow → code cut 430→100 lines

Pros & Cons

Advantages

- Modular, extensible, easy to prototype, natural language + code, human-in-loop

Limitations

- Safety risks, overhead for simple tasks, bias & privacy concerns

Related Stack

- Orchestration: LangGraph (supervisor + specialist agents for pathology, oncology, radiomics, tumor triage)
- Observability & evals: LangSmith or Langfuse (open-source) for tracing, dataset-based evals, versioning, and regression checks
- Research loop: AutoGen notebooks to experiment with agent-to-agent strategies before promoting flows into LangGraph

Key Takeaways

Objectives

- AutoGen simplifies building practical multi-agent LLM apps
- Improves performance across varied domains
- Bridges natural language workflows with programmable control

Q&A.

Thank you for listening!