Software &
Digital Platforms

AI Agent Workshop

Building AI Agents with Azure

May 05 – May 12

Contents

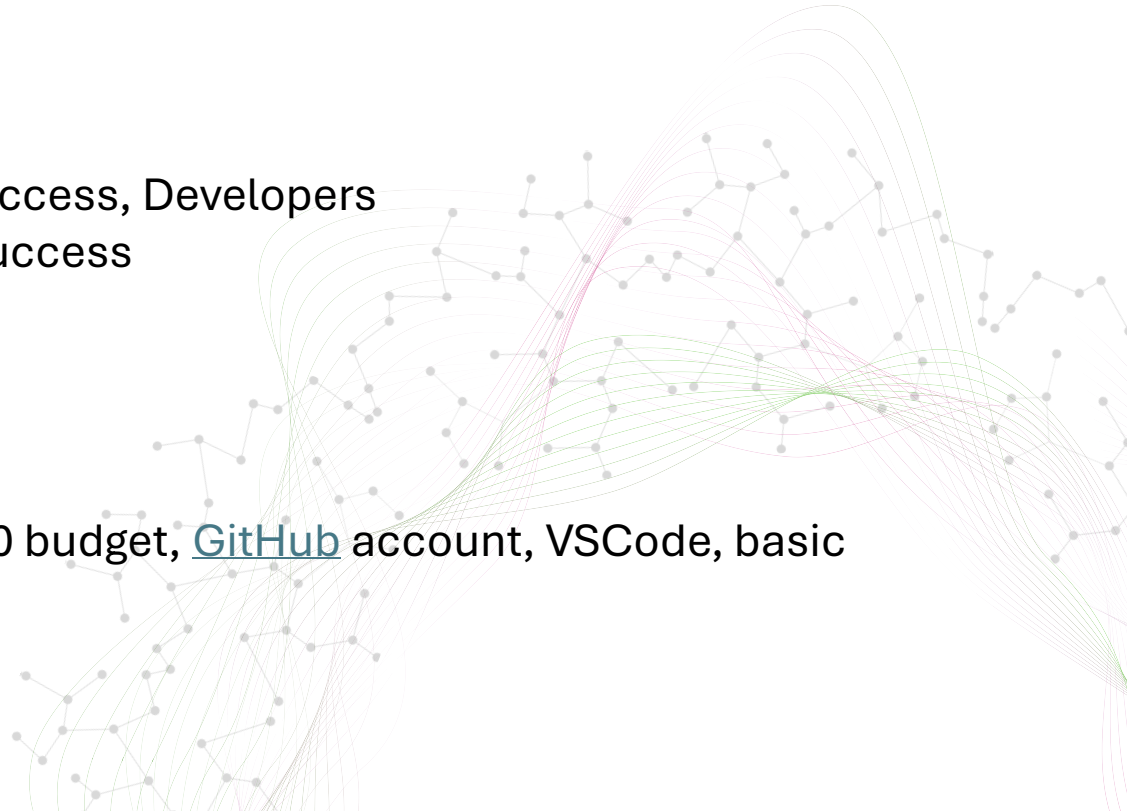
Workshop Overview

- What you will learn
  - By the end of the workshop, you will learn
    - How to build an AI Agent and explore its tools
    - What are different AI Agent Platforms and how to use them
    - Hands-on experience building single and multi-agent systems

- Getting Started
  - README

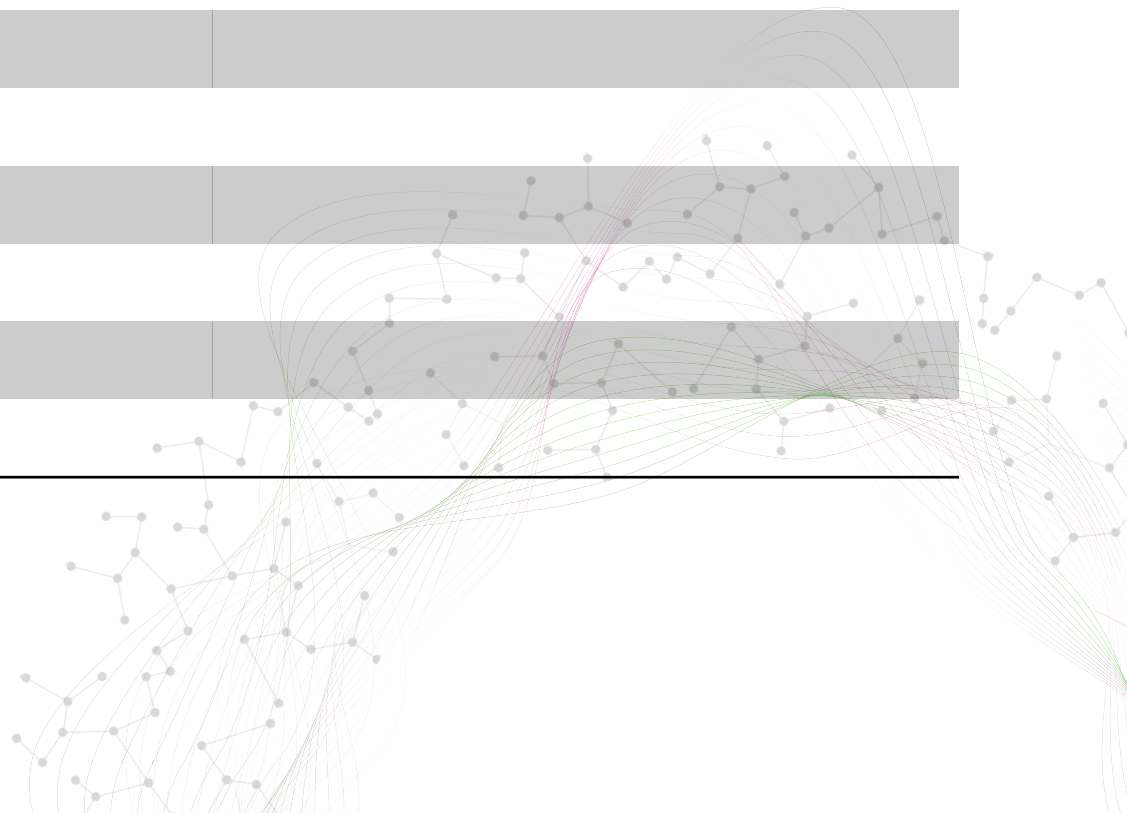
- Target Audience
  - Day1: Executives, Product Management, Customer Success, Developers
  - Day2: Developers, Product Management, Customer Success
  - Day3: Developers

- Pre-requisites
  - Day1: Basic understanding of AI and LLMs
  - Day2 & Day3: Access to an Azure subscription with $50 budget, GitHub account, VSCode, basic familiarity with python



Workshop Support Team

Name	Contact	Notes
James Nguyen		
Anil Dwarkanath		
Nicole Serafino		
Patrick O’Malley		
Heena Ugale		
Aditya Agrawal		
Claire Rehfuss		
Kirby Repko		



Day 1

Introduction to Agentic AI

What are agents?

AI designed to perform a task

Tasks can vary in level of complexity and capabilities depending on your need

Simple → Advanced

Generation

Generate summaries, images, audio, and more with an AI model and inputs.

Generally available

Retrieval

Retrieve information from grounding data, reason, summarize, and answer user questions

Generally available

Action

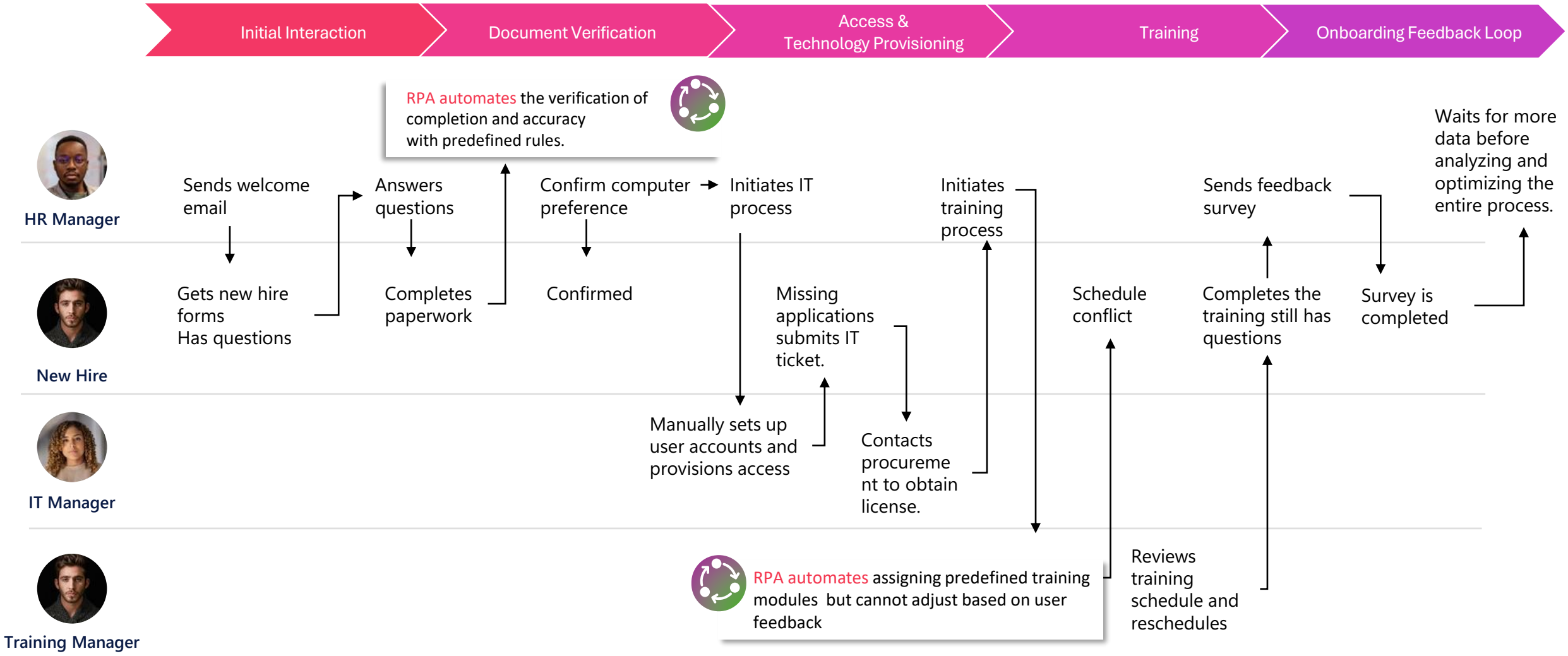
Take actions to automate workflows, and replace repetitive tasks for users

Generally available

Microsoft

TODAY – Workflow Automation

Existing solutions can only automate very specific tasks that have clear inputs and outputs



TOMORROW – AI Agents

With AI Agents, these steps can be fully automated for the first time.

Initial Interaction | Document Verification | Access & Technology Provisioning | Training | Onboarding Feedback Loop

Human in the Loop, Supervisor and Approver

HR Agent

- Provides a personalized onboarding journey based on interaction
- Assesses documents and learns from interactions
- Orchestrates the additional processes
- Gathers real-time feedback
- Identifies patterns makes informed decisions

New Hire

Fills out forms and has no questions

The new hire has questions during the training.

IT Agent

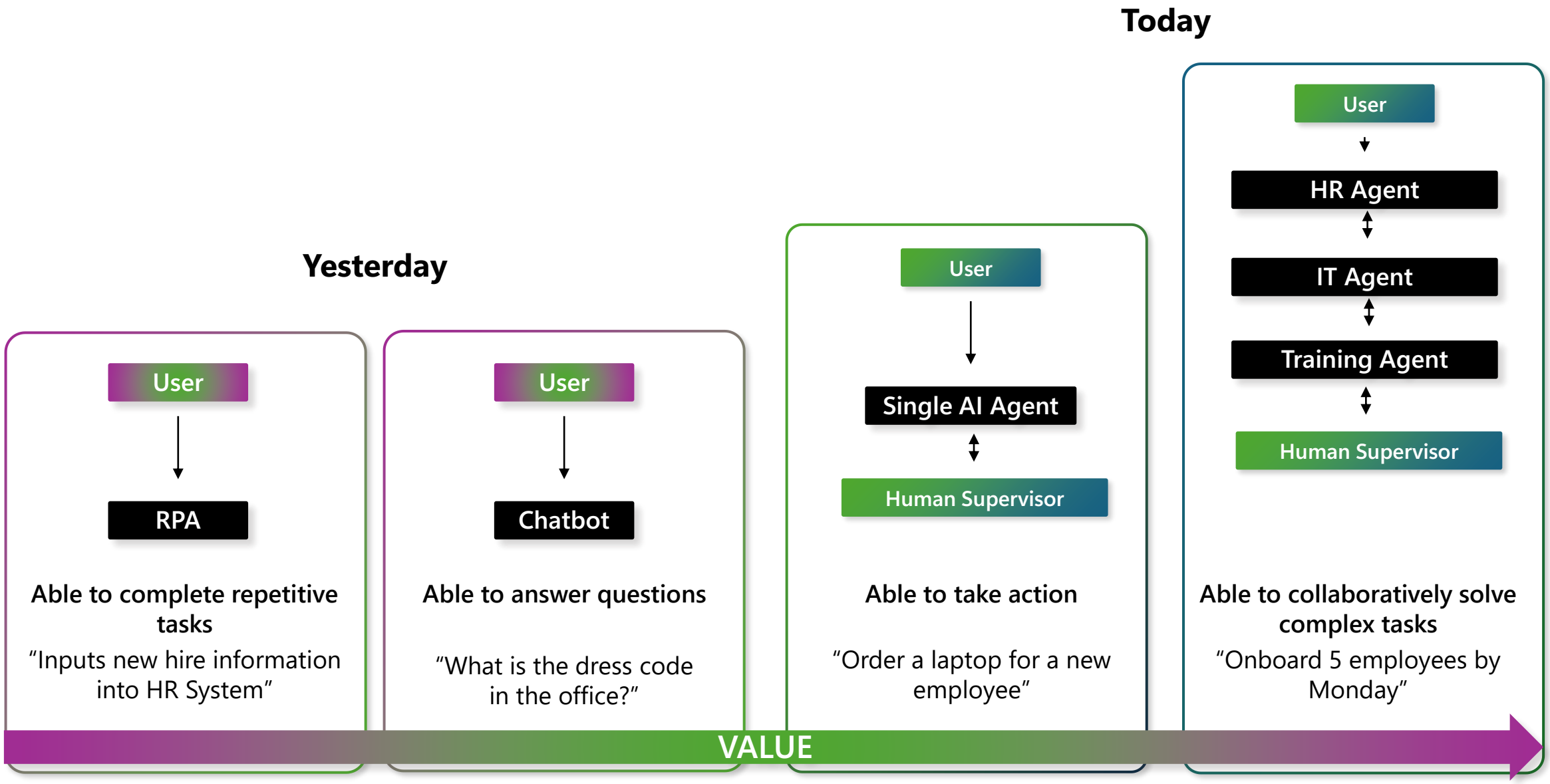
- Sets up user accounts, adapts to troubleshoot unexpected issues, and learns from errors.

Training Agent

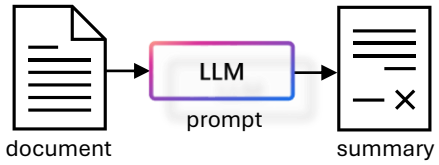
- Analyzes role, experience, and learning preferences to recommend training

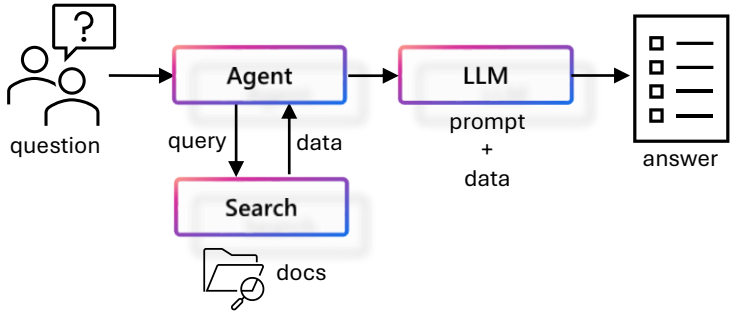
Provides adaptive training

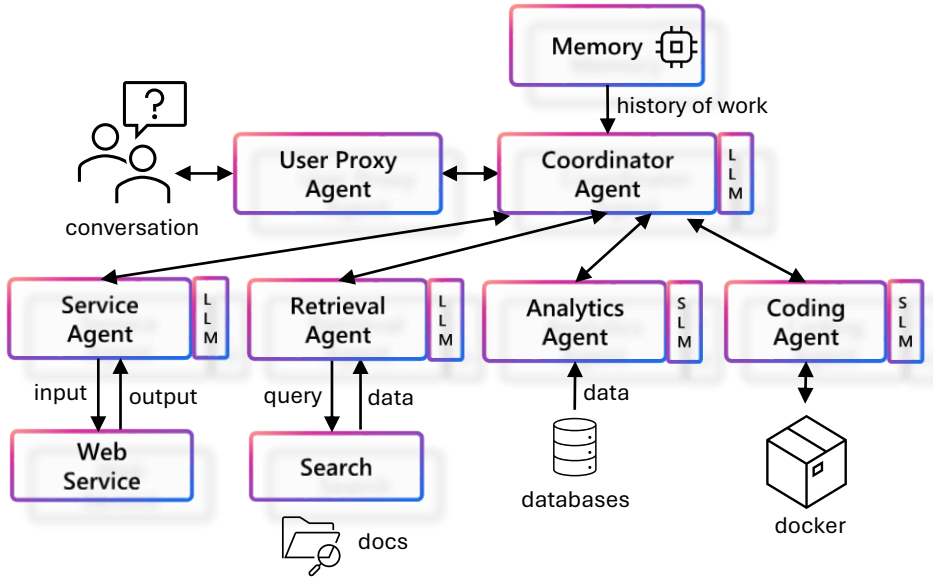
Promising even more efficiency, value, and advantage



Spectrum of LLM-based Solutions







No Agent

Narrow one shot task

Ex: log to JSON

Single Agent

Clearly scoped iterative task

Ex: providing an answer with supporting evidence to a complex question

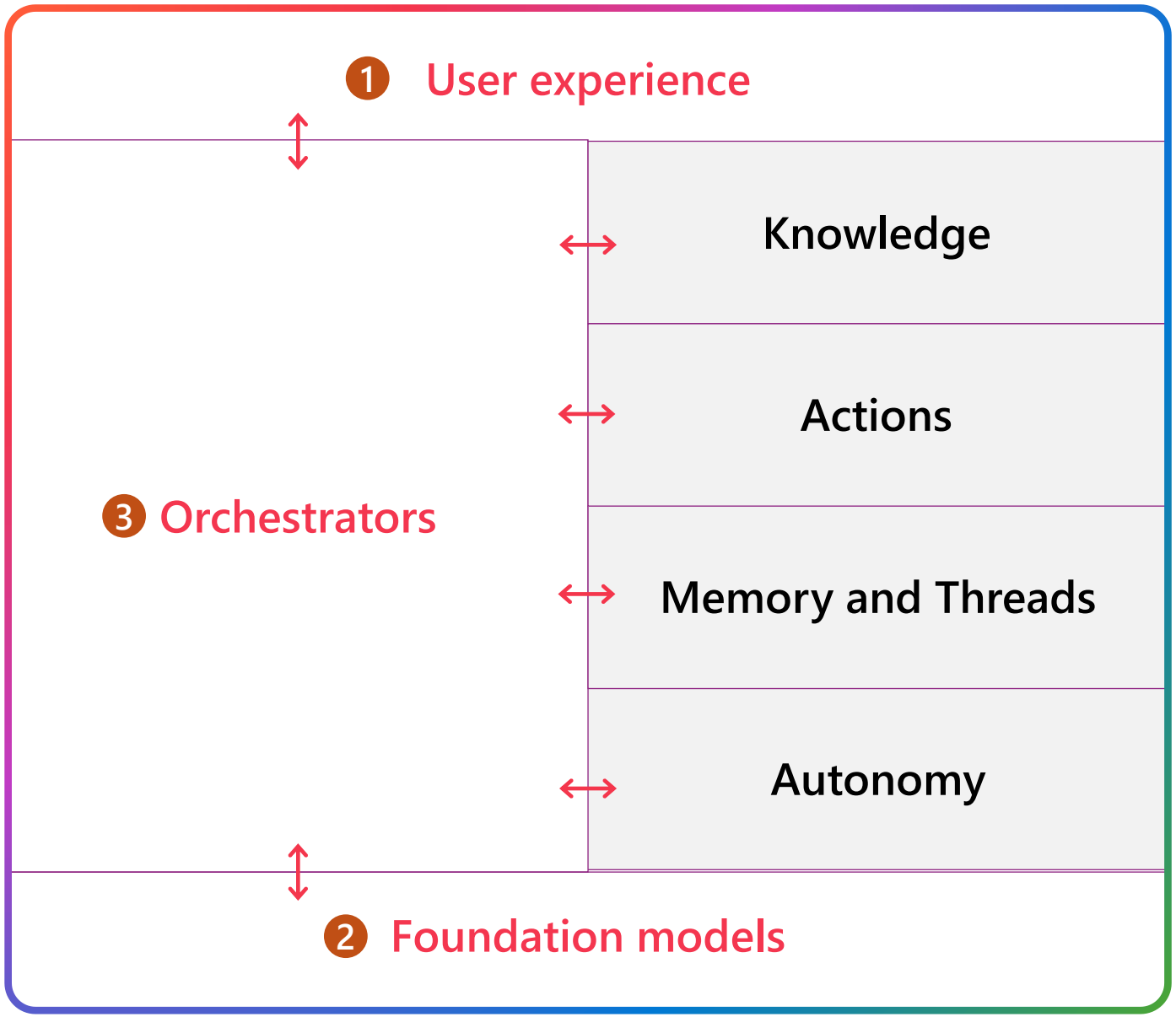
Multi-agent Systems

Wide scope complex use case requiring diverse skills

Ex: Propose 2 Instagram marketing campaigns including assets that would leverage the top 2 recent trends in our past quarter US Sales to boost our mailing list user base and predict the impact of each campaign

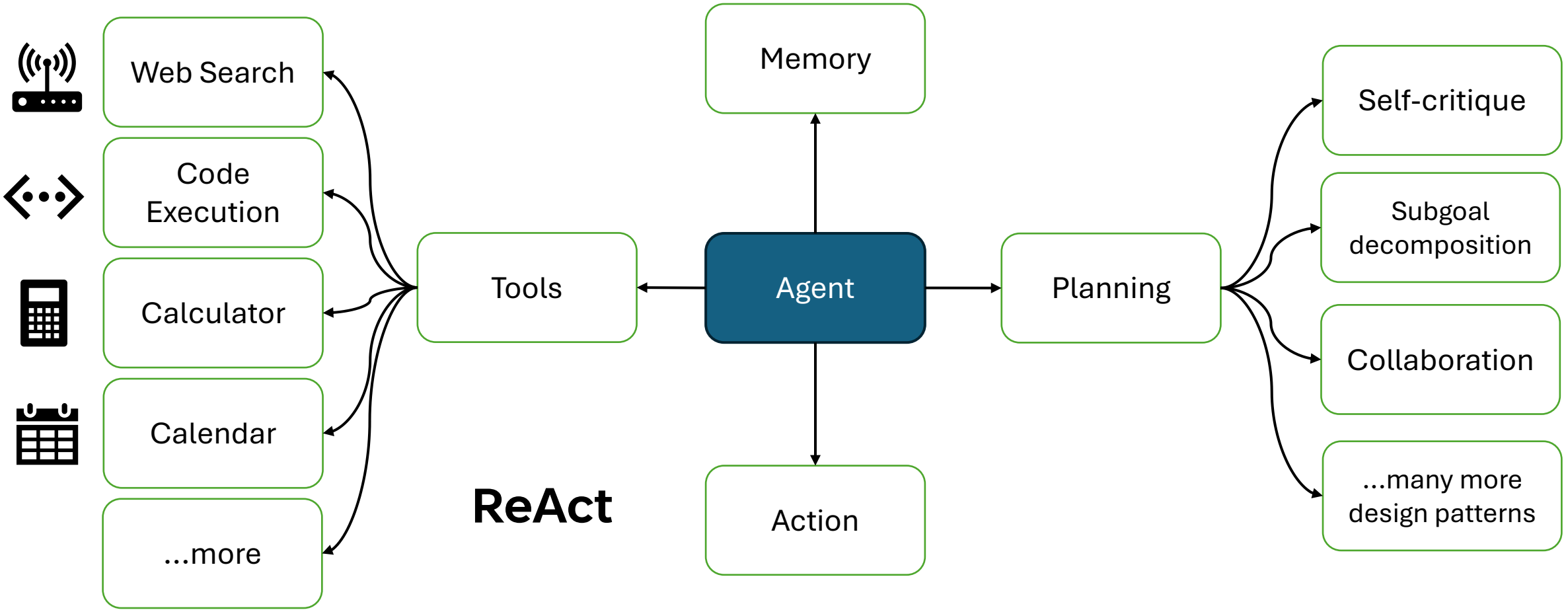
VALUE

Components of AI Agents



1. **User Experience**: The prompt that starts the whole process of the agent execution, as well as the human-agent interaction.
2. **Knowledge**: Search and retrieve information from online sources or company knowledge base, to ground the models.
3. **Actions**: Helps the agents perform certain actions (e.g. send an email, write a report) through connections to key applications.
4. **Memory and Threads**: Captures and stores the past interactions for hyper-personalisation and increased human-like interactions
5. **Autonomy**: Leaves the agents perform the tasks through an event-driven, trigger-based approach.
6. **Foundation Models**: Enables AI Agents to think throughout the process, helping to plan and reflect.
7. **Orchestrators**: Whether to be the client-side code or orchestrate across multiple agents across multiple cloud, orchestrators are key to bringing everything together.

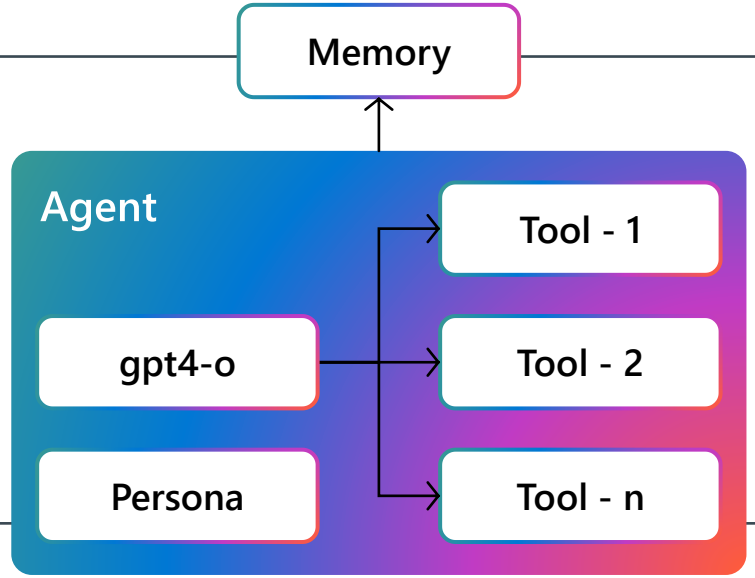
Agentic AI capabilities



Agent Abstractions - Agent First-Class Citizen

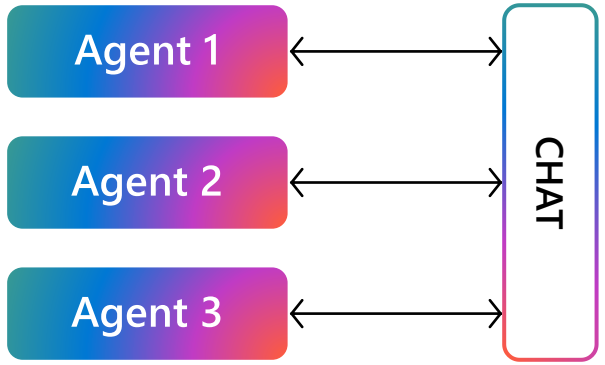
Agent as high-level abstraction

- **LLM** (gpt4-o, o1 etc.)
- **Persona** (system prompt)
- **Tools** (function code calls)

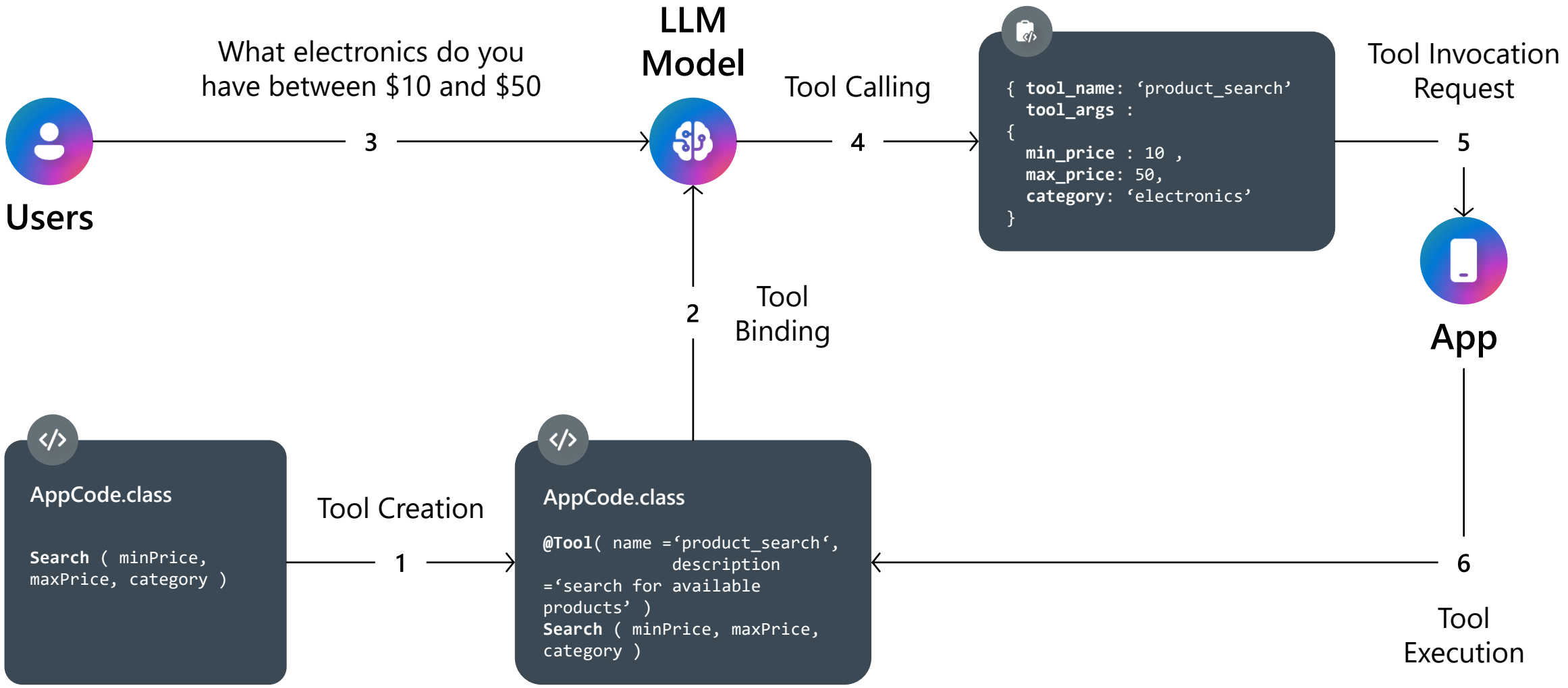


Agent Chat as layer for collaboration

- **Multiple agents** can engage with each other
- Enables multi-turn or parallel execution



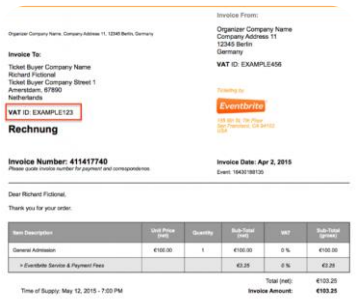
Agentic Pattern - Tools Calling



Agentic Pattern - ReAct Planning with Tools Calling

Users

Pay this bill for me



Bill abc123 successfully paid

Payment Agent

Tools:

- scanImage (filename)
- transactionHistory (billId)
- paymentService (billId, Amount, Payee)

Instructions:

- You are a home banking assistant allowing users to pay the bill uploading a picture
- Always check if a bill has already been paid before submitting a payment
- Confirm the payment result

Planning

while (new tools execution request)

1. **scanImage**
   (bill-abc123.jpeg)
   → abc123, 100$, payeename
2. **transactionHistory**
   (abc123)
   → no results
3. **paymentService**
   (abc123, 100$,payeename)
   → ok

Agentic Pattern - Memory

Short Term

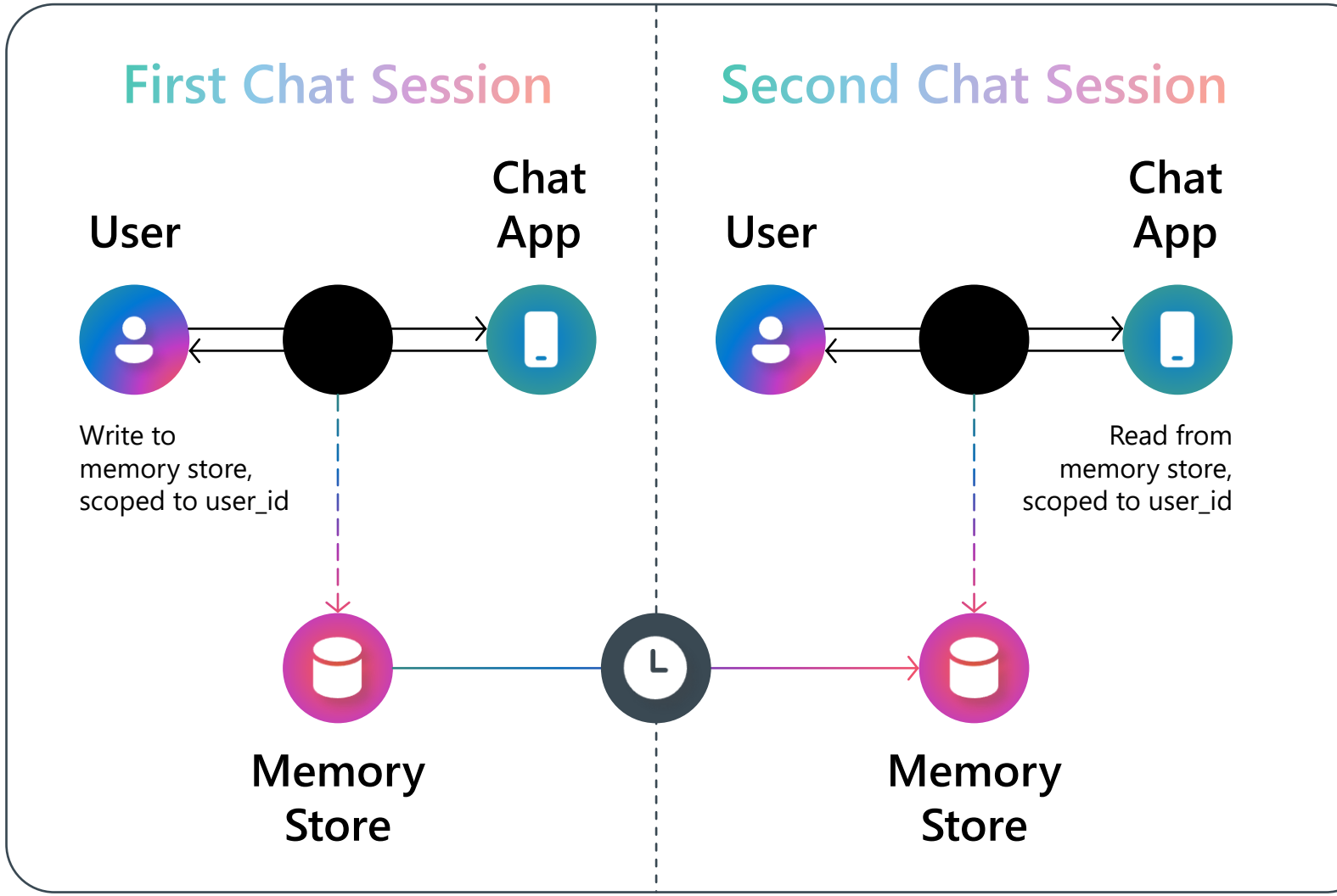
- Access steps info in one loop iteration
- Shared state context
- Chat history

Long Term

- Access steps info in long running conversation
- State persistence

Conversation History Truncation

- Trim by tokens
- Trim by message count
- Trim + summary (LLM call required)

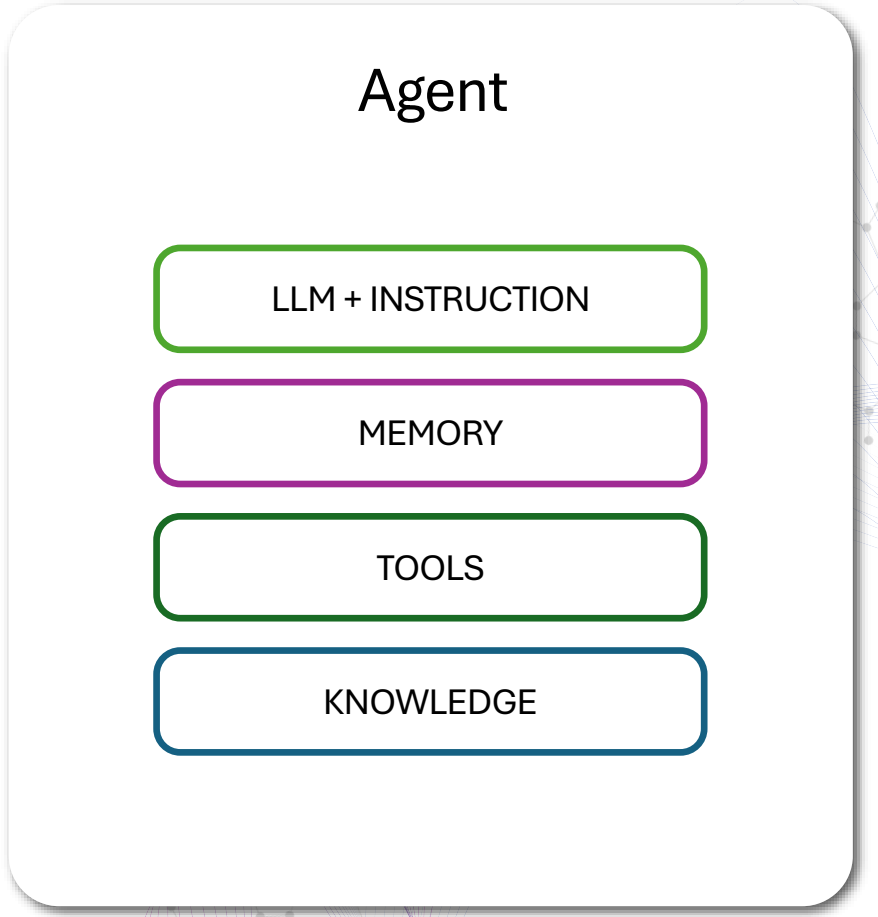


Memory - Providing Context for Agent

Memory is a **foundational** capability that allows agents to store and leverage past interactions to deliver personalized, context-aware experiences and enhance workflow efficiency.

Definition

- Memory store – the place of storage for information across multiple threads
- Memorizing – act of taking things from a thread and storing it elsewhere
- Recalling – the act of retrieving information from memory

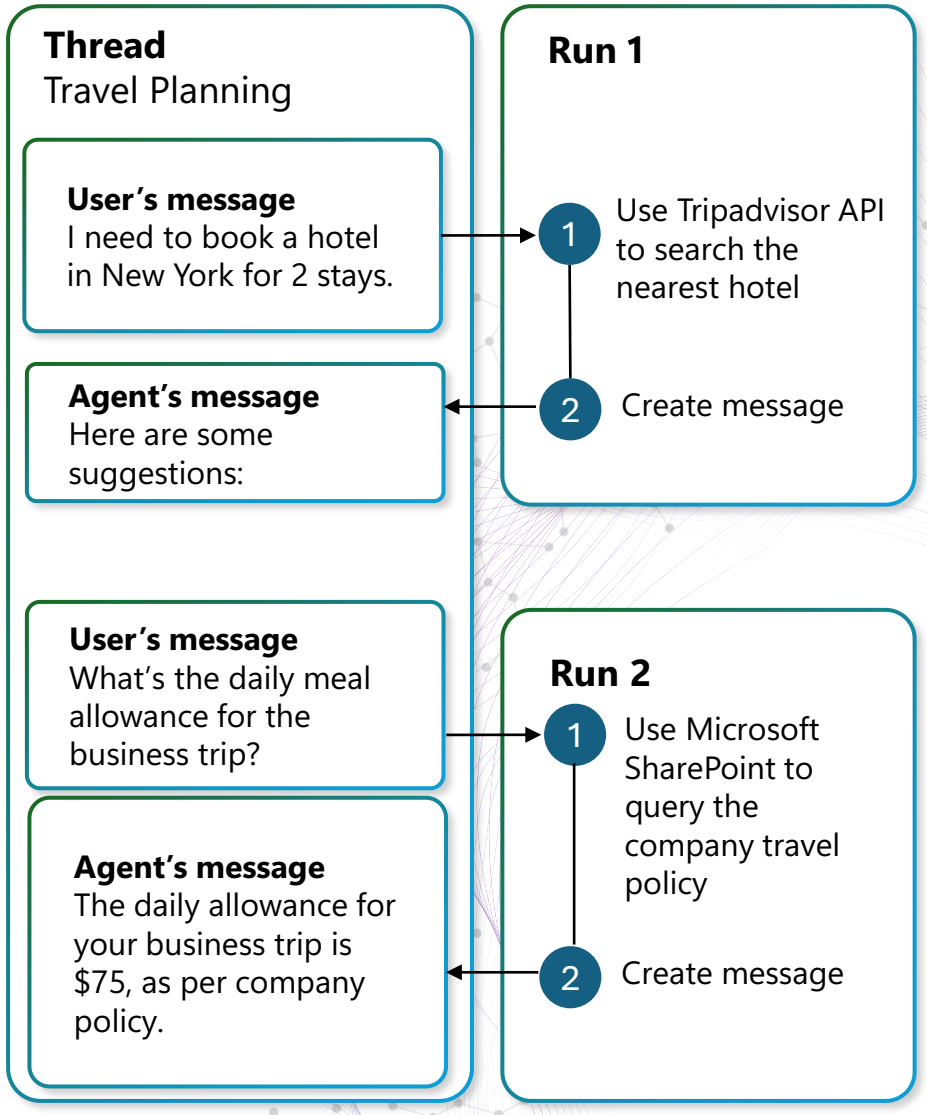


Thread – Providing History to Conversations

Threads store conversation history. Currently, agents store threads in a managed, multi-tenant Cosmos DB. Soon, customers will be able to bring their own (BYO) Cosmos DB as a part of the standard setup, ensuring all thread messages are stored in their customer tenant for enhanced security.

Definitions

- Thread – a conversation session between an agent and a user. Threads store Messages and automatically handle truncation to fit content into a model's context.
- BYO Cosmos DB – an enterprise feature that provides secure, single-tenant data control, allowing you to maintain complete control over your customer data.



Thread Storage VS Memory

	Thread Storage	**Memory Store**
Description	CRUD ALL thread metadata, orchestration context, and conversation state in a scalable, consistent **operational** storage account	Store information about previous conversations in an index with intelligent **retrieval** capability
Requirements	Consistency, scalability for operational interactions Real-time performance	Indexing and chunking data for vector/semantic search and retrieval

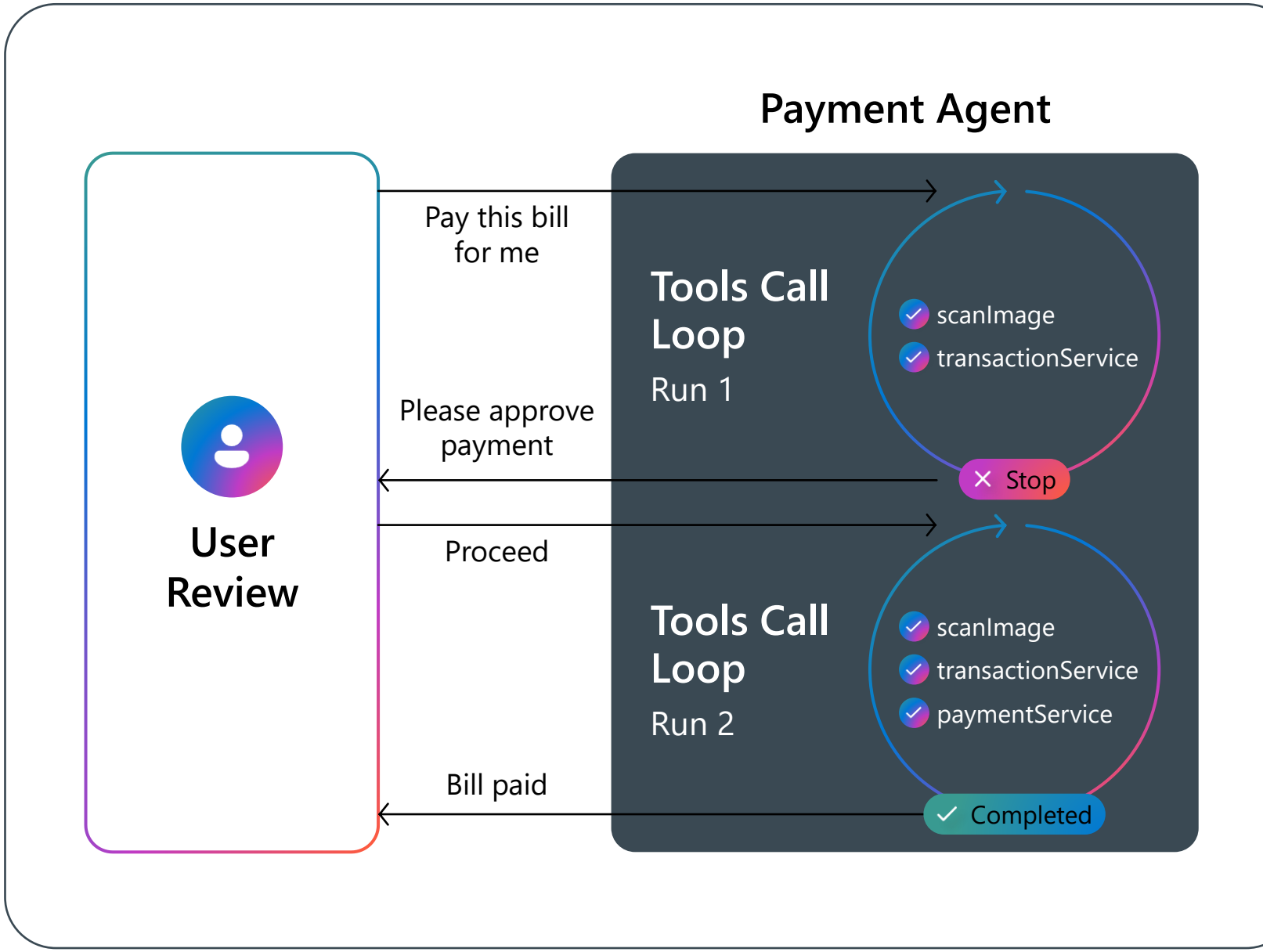
Agentic Pattern - Flow control

Looping Termination

- MaxIterations
- Message termination
- Human step /Human in the loop

Human in the loop

- Action execution approval
- Escalation
- Data review



Day 1

Agentic System Design Patterns

Single Agent	Multi-domain	Multi-agent

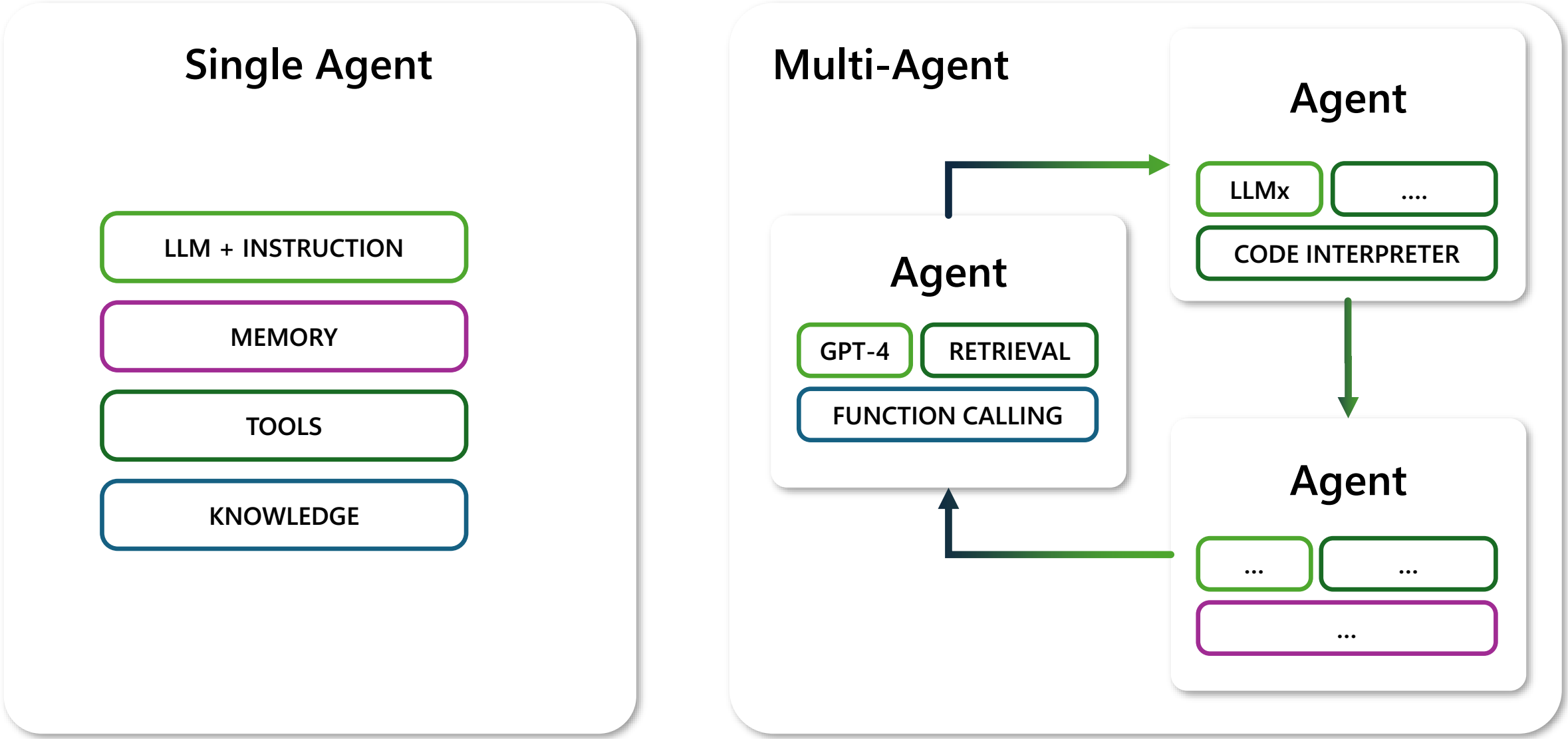
Design Patterns Explored in Workshop

Intelligent Single Agent

Multi-domain Multi-agent

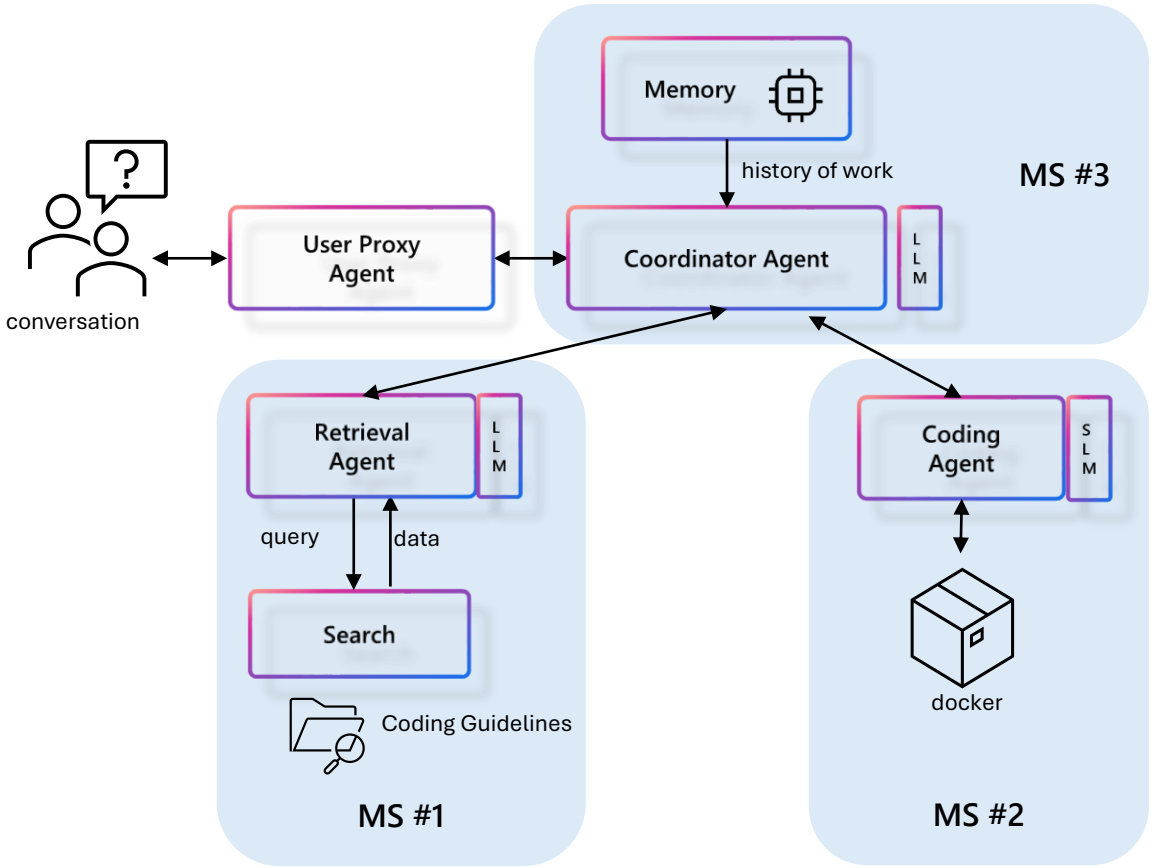
Collaborative Multi-Agent

Single vs Multi-agent



Multi-Agent System

A complex problem is decomposed into smaller, manageable parts, each addressed by specialized agents, effectively a micro-service (MS). These agents work together in a coordinated manner within a workflow to efficiently solve the overall problem.



Critical Design Elements

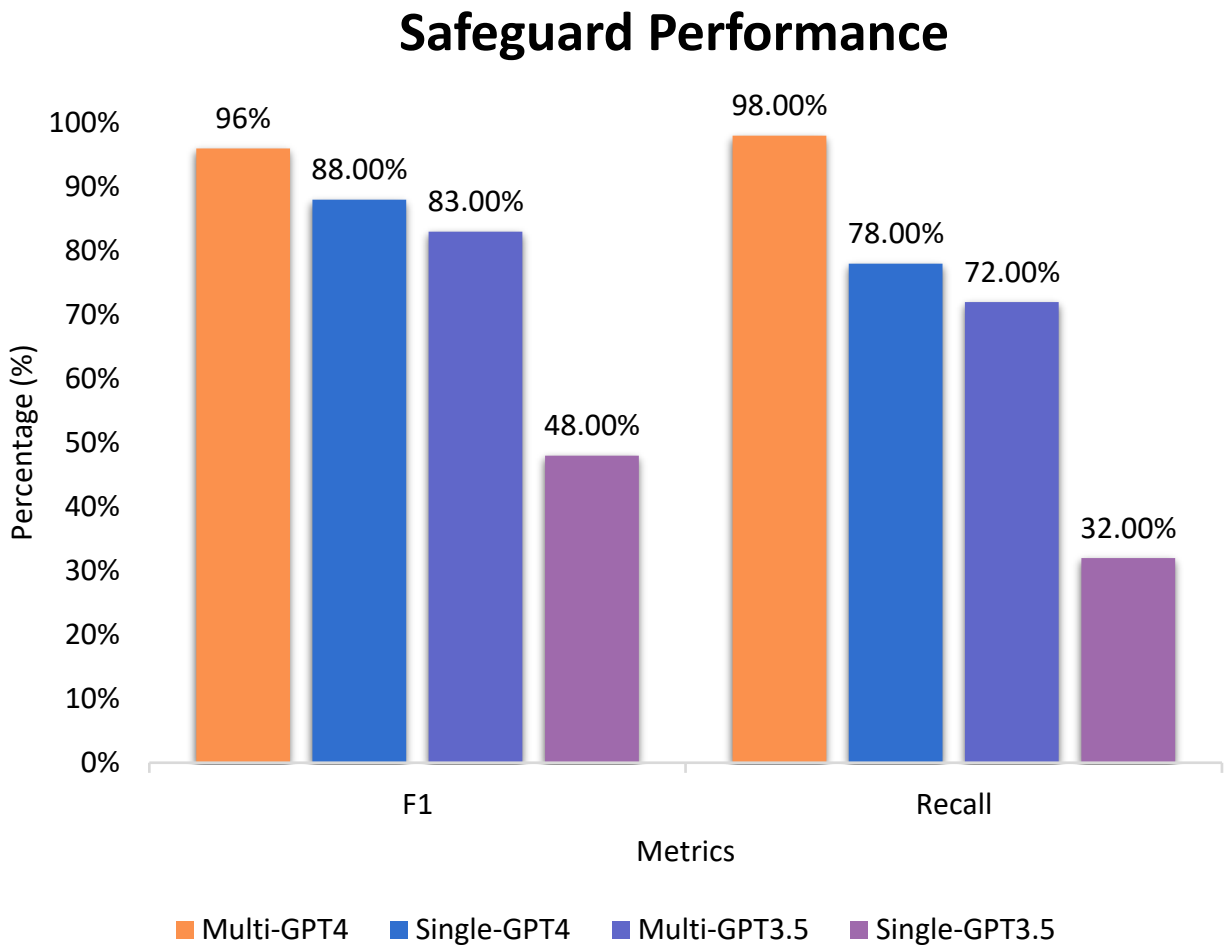
- Adaptive planning within scope of existing tightly scoped skills (agents)
- Handles ambiguity by discussing and refining requirements with human
- Memory to handle complex long running execution of a plan
- Effective inter agent communications
- Test, monitor, release & maintain each agent independently to quickly handle quality & safety issues

When should I use multi-agents over a single agent?

Multi agent systems can solve more complex tasks

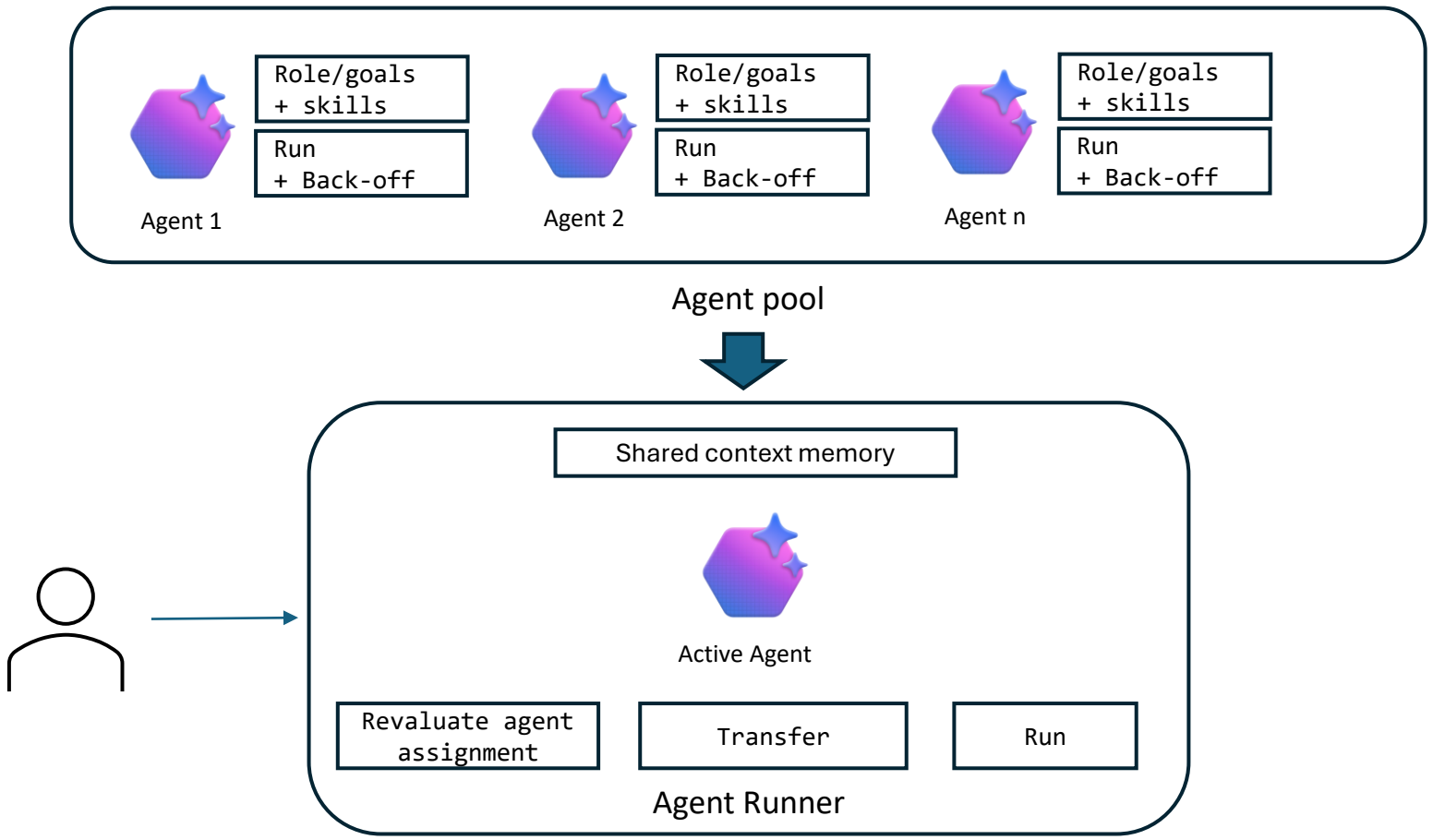
Single agents can solve a wide range of problems and are simpler to implement.

Multi-agent systems should be used
- if a single agent is unable to solve the challenge
- to handle tasks that involve more data, diverse roles, or complex workflows



Multi-Domain Multi-Agents System

Multiple domain-specific agents are orchestrated by an Agent Runner to scale across multiple domains while appearing as a single agent to users.



Critical Design Elements

- Agents capability descriptors
- Scalable Agent Runner able to manage 10s to 100s of agents
- Ability to manage domain switching with proper memory management
- Avoid single interceptor problem as individual agents maintain direct communication with user and can hand off when needed

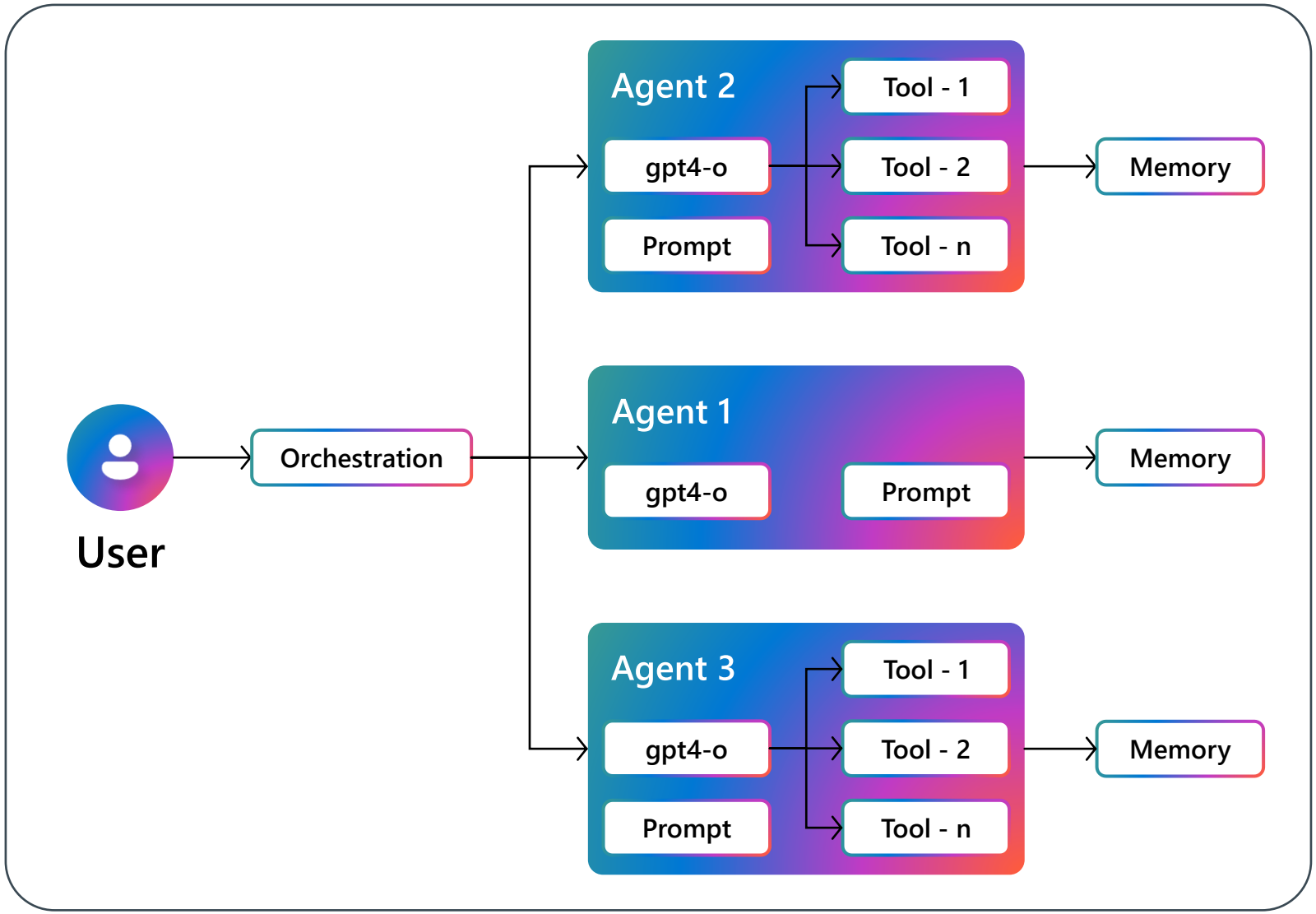
Multi Agent Logical Architecture

Each agent is specialized in different tasks or aspects of a problem

Agents can communicate and coordinate with each other. Structured orchestration is crucial

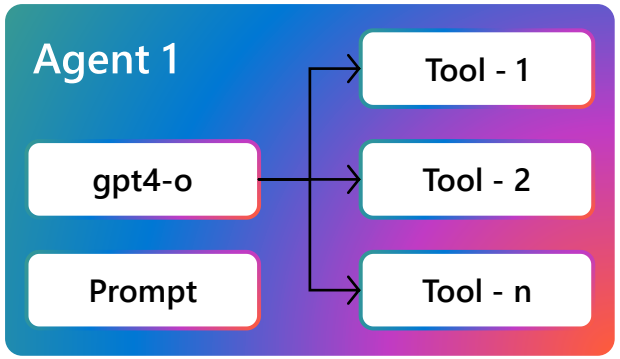
2 primary categories based on orchestration types

- Vertical Architecture
- Horizontal Architecture

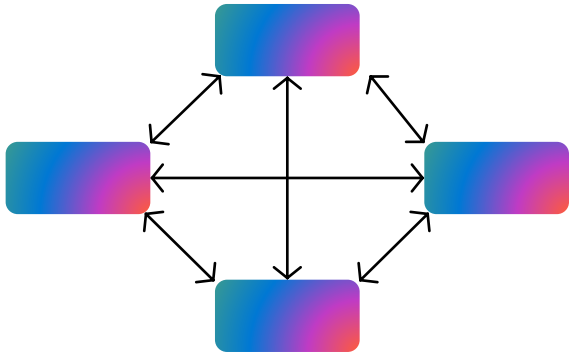


Agents orchestration and communication styles

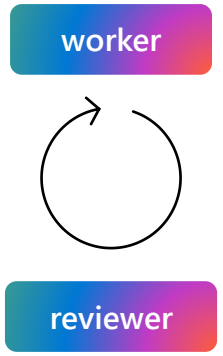
Single Agent



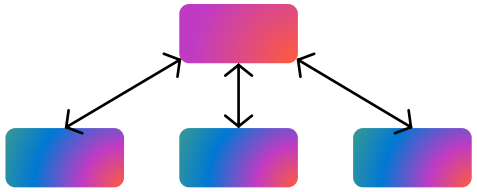
Network



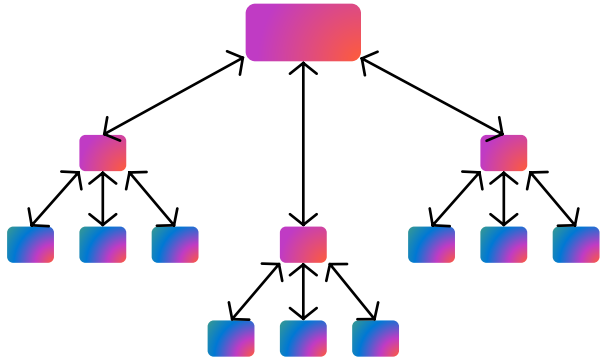
Reflection



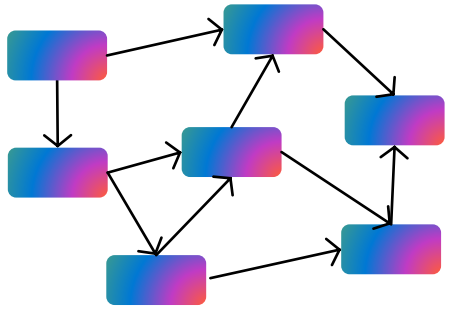
Supervisor



Hierarchical



Custom



Multi-Agent Collaboration

- **Specialization**
  - Different agent are configured for specific tasks. They can tackle different aspects of a complex problem.
  - Multiplying the power of a single agent.
- **More modular and easier for developers**
  - Keeping the system easy to maintain and add or remove components.
  - Increasing collaboration across different teams' Copilots.

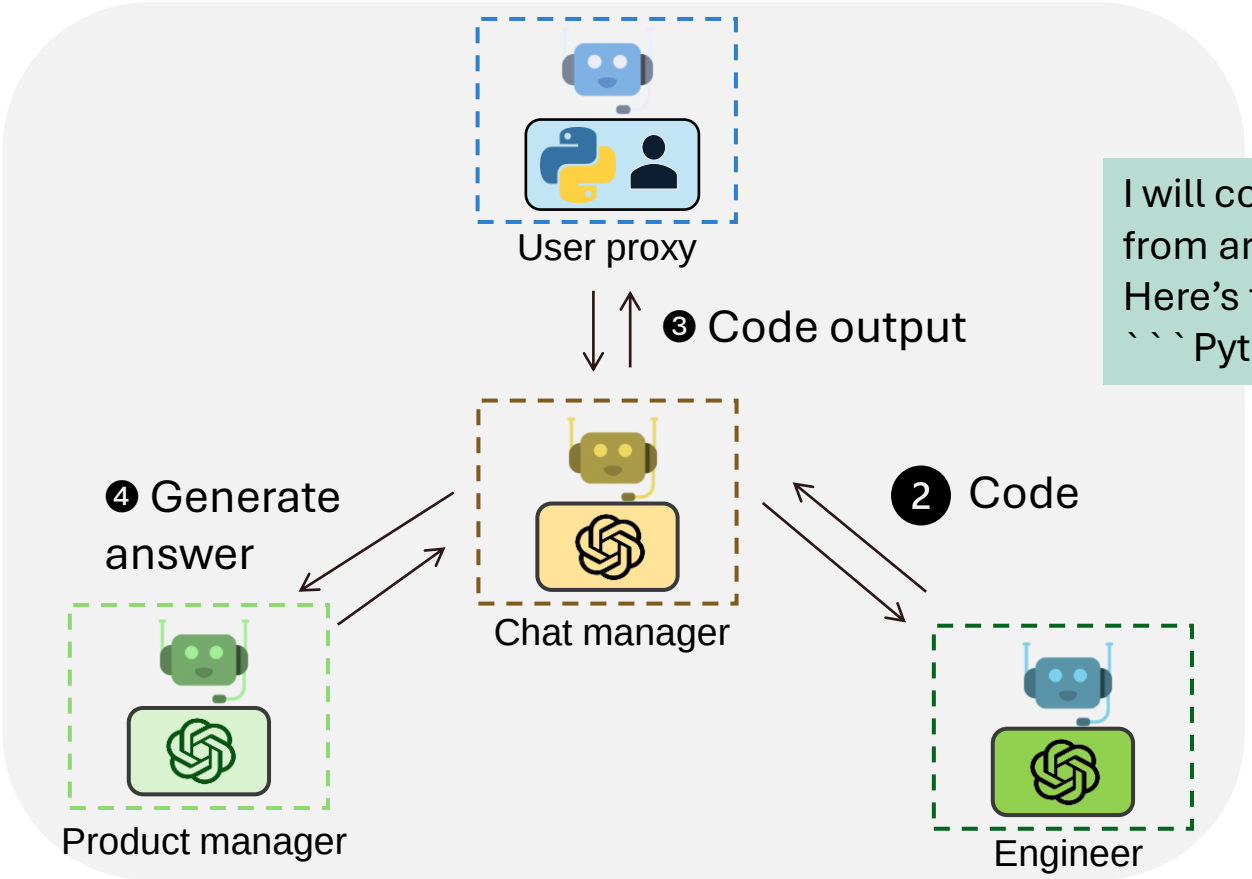
Find latest papers about gpt-4o on arXiv and find their potential software applications.

User

1 Question

5 Final answer

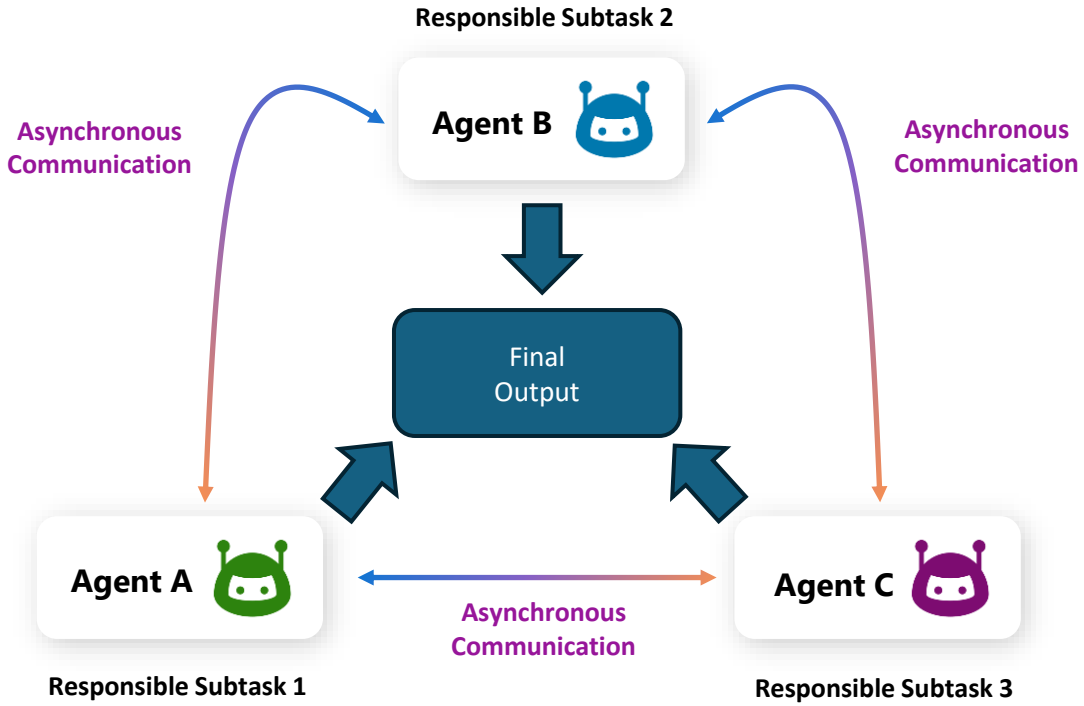
I found several recent papers on GPT-4o from arXiv. Here are the most relevant ones along with their potential applications in software:



I will collect the latest papers from arXiv related to GPT-4o. Here's the code to do that:
```Python code```
```

Cooperative vs Competitive Agents

Cooperative based-learning

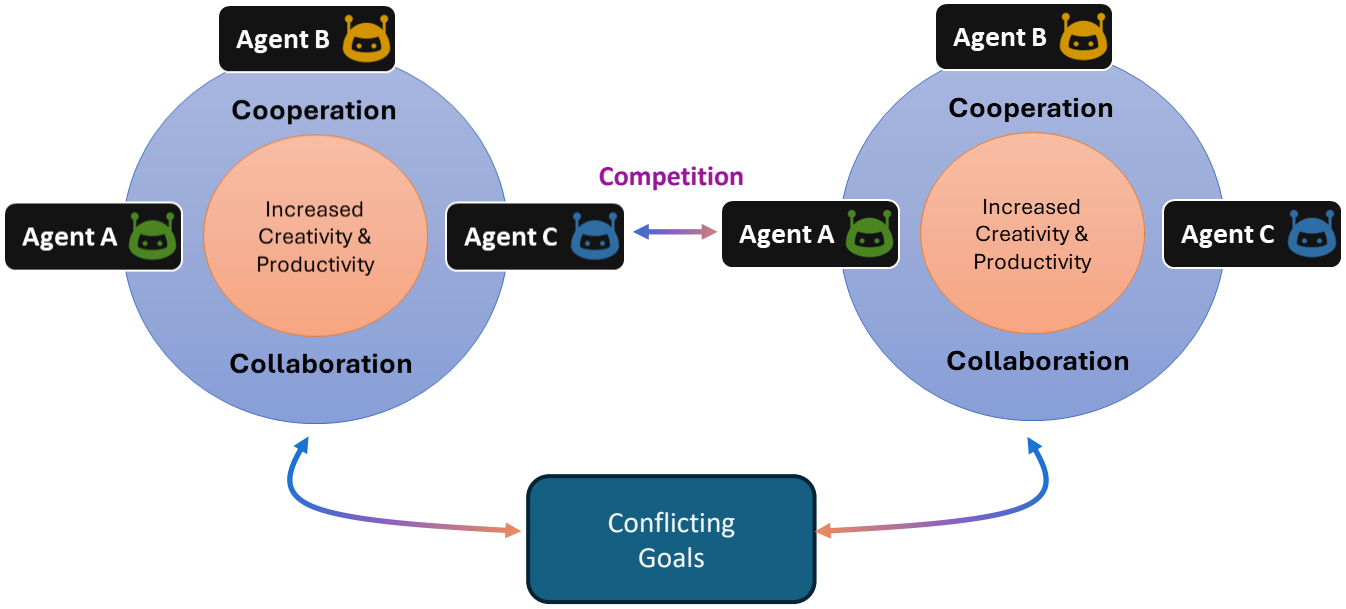


Cooperative agents work together towards shared goals, enhancing problem-solving capabilities and efficiency through collaborative efforts, characterized by trust and open communication.

Complex Problem Solving | Efficiency | Adaptability

Competition based-learning

Seems not useful in a machine scenario



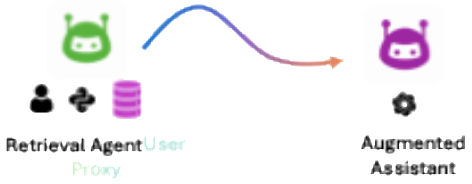
In contrast, competitive agents operate with individual goals in adversarial settings, focusing on optimizing their own outcomes, often with strategic communication and planning.

Resource Allocation | Decision Making | Realism

Multi Agent Examples

Student-Teacher

RAG



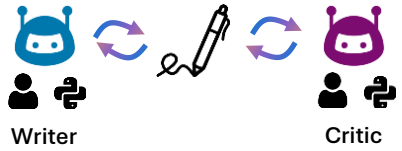
Decision Making

Math & Logic

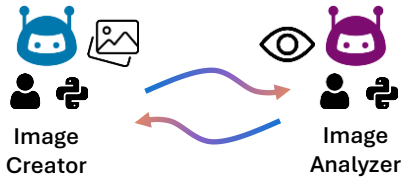
Task Solving

Conversational

Reflection



Multimodal



Day 1

Agentic Frameworks

Azure AI Agent Service	Semantic Kernel	Autogen

Role of Frameworks

Abstraction Layers

Reduce Complexity of integrating tools and workflows

Built in Functionality

Common tools, memory management, decision making

Best Practices

Streamline development process

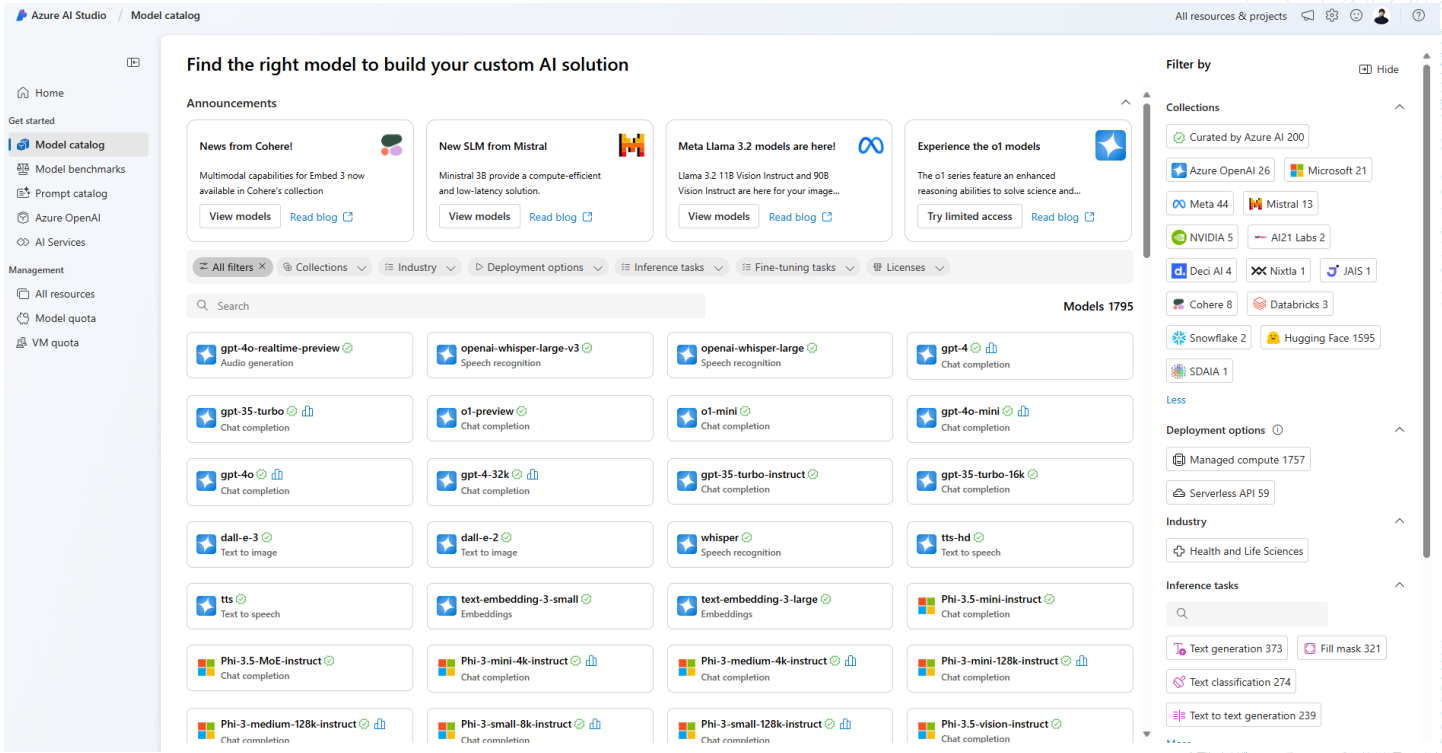
Azure AI Foundry Enables Flexible Model Selection

Azure OpenAI Service models

Models as a Service (Serverless API)

Fine-tuned models

Multi-modal support



Azure AI Agent Service

Azure AI Agent Service

Trust

Customer control over data, networking, and security

- BYO-file storage
- BYO-search index
- BYO-virtual network
- BYO-thread storage
- Tracing/monitoring
- Evaluation

Choice

Model choice and flexibility with the model catalog

Azure OpenAI Service

o3-mini, o1, GPT-4o, GPT-4o mini

Models-as-a-Service

- Llama 3.1-405B-Instruct
- Mistral Large, Small
- Cohere-Command
- DeepSeek v3

Skills

Richest set of enterprise connectivity

Knowledge

Actions

Logic Apps* Azure functions OpenAPI

Azure AI Foundry portal

Azure AI Foundry SDK

AIAS Enterprise Readiness

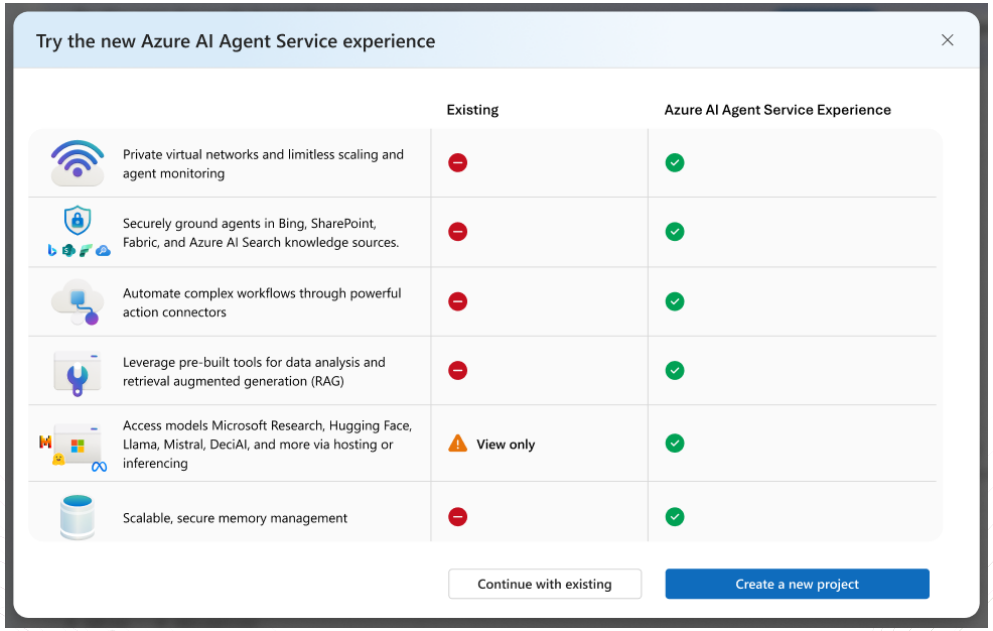
Bring your own storage

Keyless setup and authentication

Private Virtual Network support

Tracing/ monitoring

Content filters

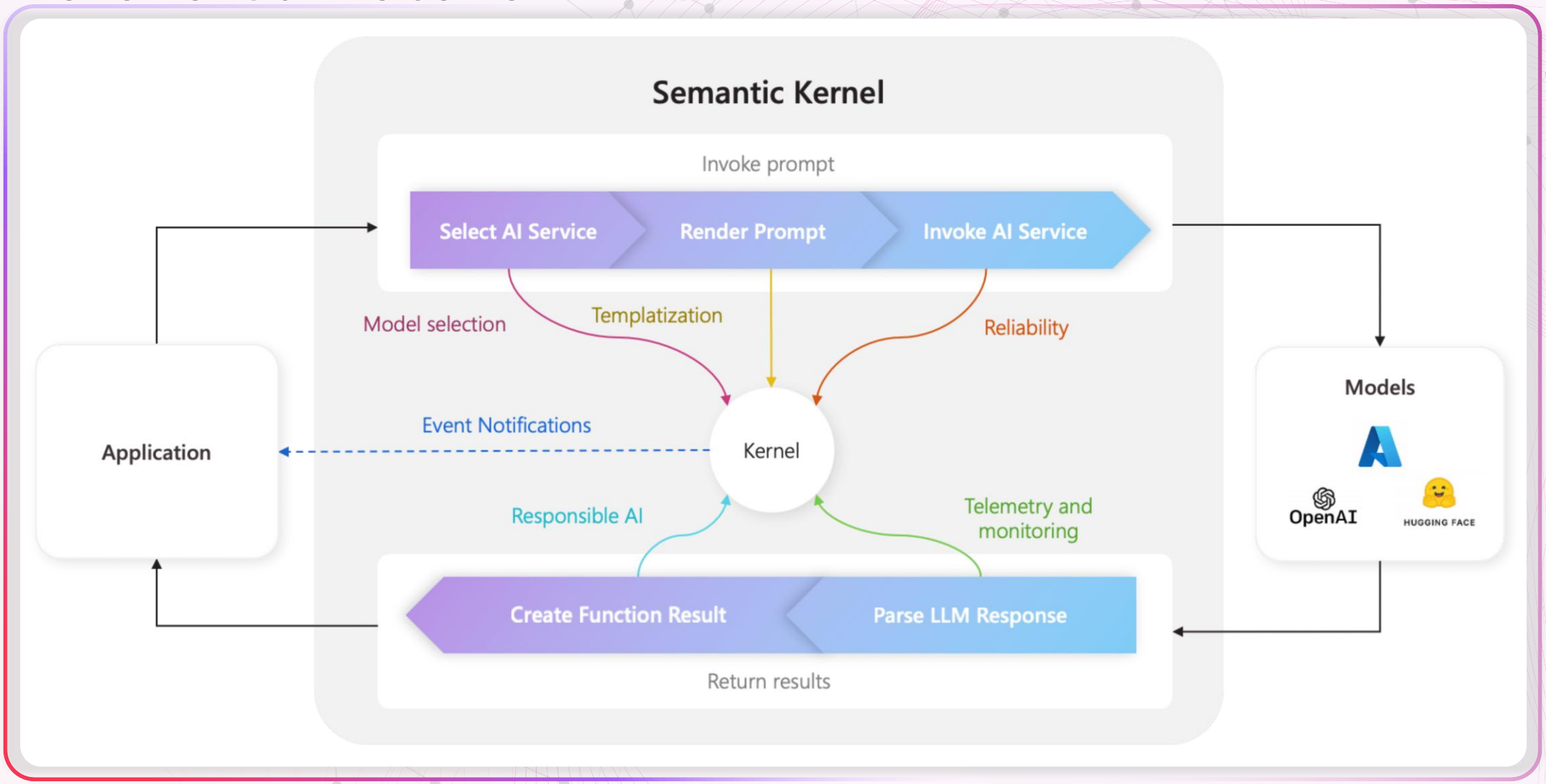


Semantic Kernel

Semantic Kernel is lightweight, **open-source**, production-ready, orchestration middleware that lets you easily add AI to your apps.

- Full SDK designed to build AI agents with ease, excellent for single agents and can be extended for multi-agents with integrations to AutoGen
- Extensible and compatible with models LLMs or SLMs.
- Ideal for developers looking to leverage AI orchestration patterns similar to those used in Microsoft's Copilot systems in their own applications

The kernel is at the center



Semantic Kernel Agents

Reflection multi-agent

- Coding agents
- Digital content creation agents

ChatCompletionAgent

- Name
- Instructions
- Kernel (Services, Plugins)

Reflection loop with 2 agents

- Blog post writer
- Blog post reviewer

SK sample location

```python
from agents.base_agent import BaseAgent
from semantic_kernel.agents import ChatCompletionAgent, ChatHistoryAgentThread
from semantic_kernel.connectors.ai.open_ai import AzureChatCompletion
from semantic_kernel.connectors.mcp import MCPSsePlugin
```

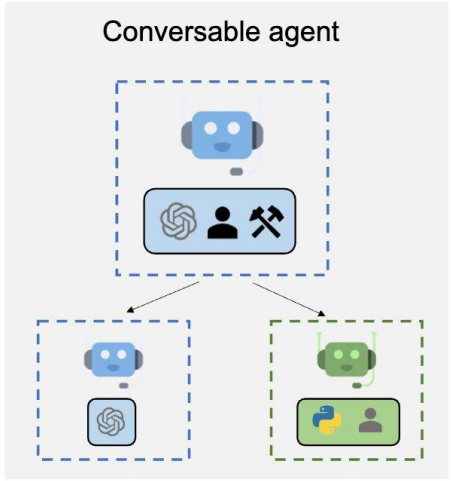
```python
async def chat_async(self, prompt: str) -> str:
    # Ensure agent/tools are ready and process the prompt.
    await self._setup_agent()

    response = await self._agent.get_response(messages=prompt, thread=self._thread)
    response_content = str(response.content)

    self._thread = response.thread
    if self._thread:
        self._setstate({"thread": self._thread})

    messages = [
        {"role": "user", "content": prompt},
        {"role": "assistant", "content": response_content},
    ]
    self.append_to_chat_history(messages)

    return response_content
```

Autogen

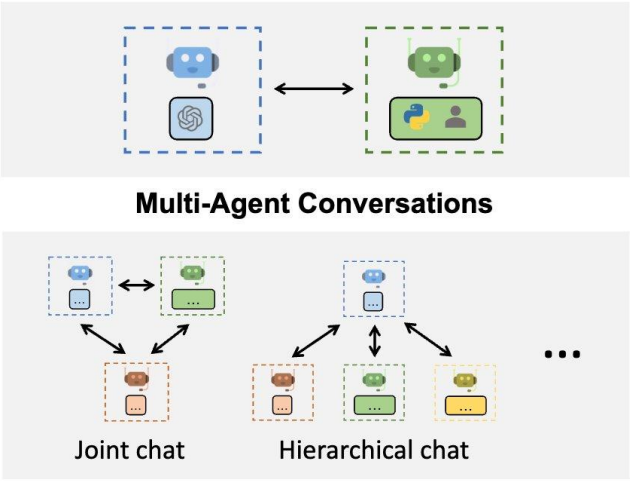
- Powerful multi-agent framework with prebuild conversation orchestration patterns for handling complex agent systems
- Extensible and compatible with models LLMs or SLMs.
- Powered by collaborative **research studies** from Microsoft, Penn State University, and University of Washington.
- AutoGen simplifies the orchestration, automation, and optimization of a complex LLM workflow.

AutoGen concepts

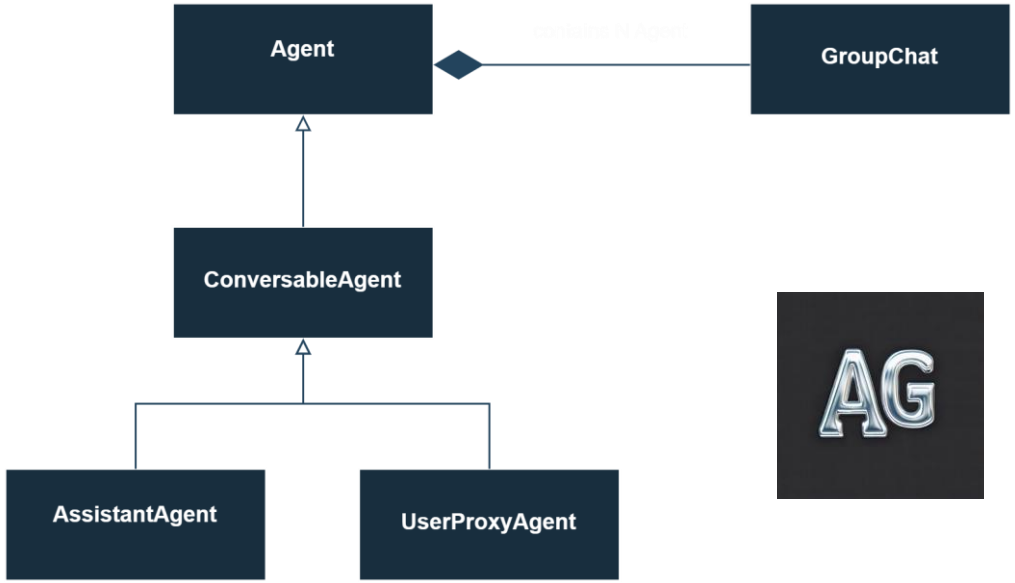
- **Customizable and conversable agents**:
  AutoGen uses a generic design of agents that can leverage LLMs, human inputs, tools, or a combination of them
- **Conversation programming**:
  - Defining a set of conversable agents with specific capabilities and roles
  - Programming the interaction behavior between agents via conversation centric computation and control.



Agent Customization



Flexible Conversation Patterns



[AutoGen | AutoGen (microsoft.github.io)]