LLMs and Agents

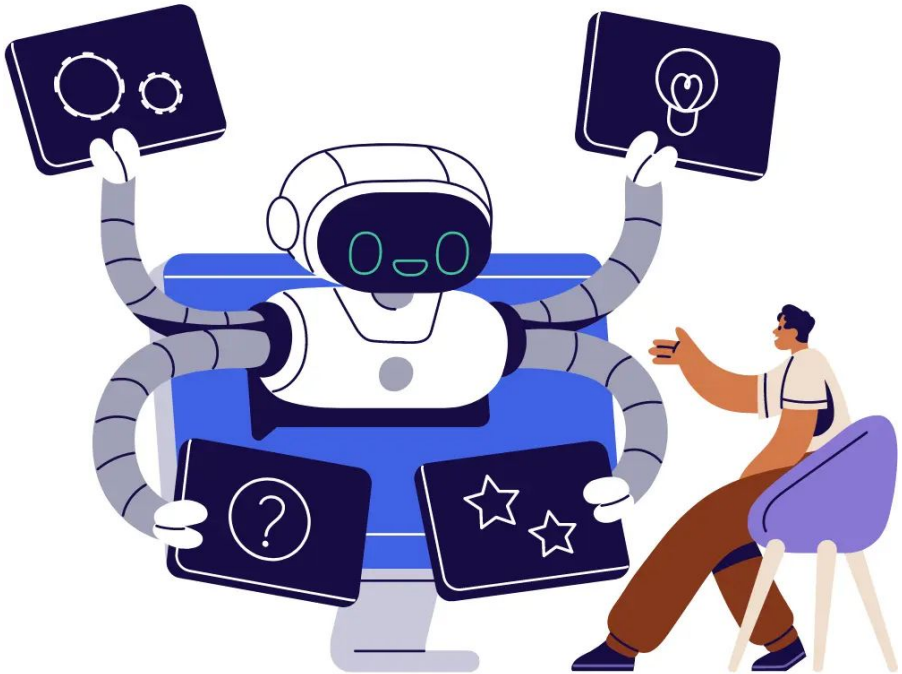
Where the Magic is Amplified

Vasudeva Varma

LLM Agents: Autonomous systems designed to achieve goals by observing, reasoning, and acting upon the world using tools.

Key Features:

- **Autonomy**: **Act independently** without human intervention.
- **Proactivity**: Plan and execute tasks to reach goals, even **without explicit instructions**.
- **Adaptability**: **Integrate logic and reasoning** with external information for dynamic tasks.



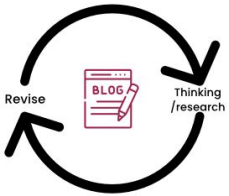
Non-agentic workflow

(zero-shot) :

Please type out an essay on topic X from start to finish in one go, without using backspace.

Agentic workflow:

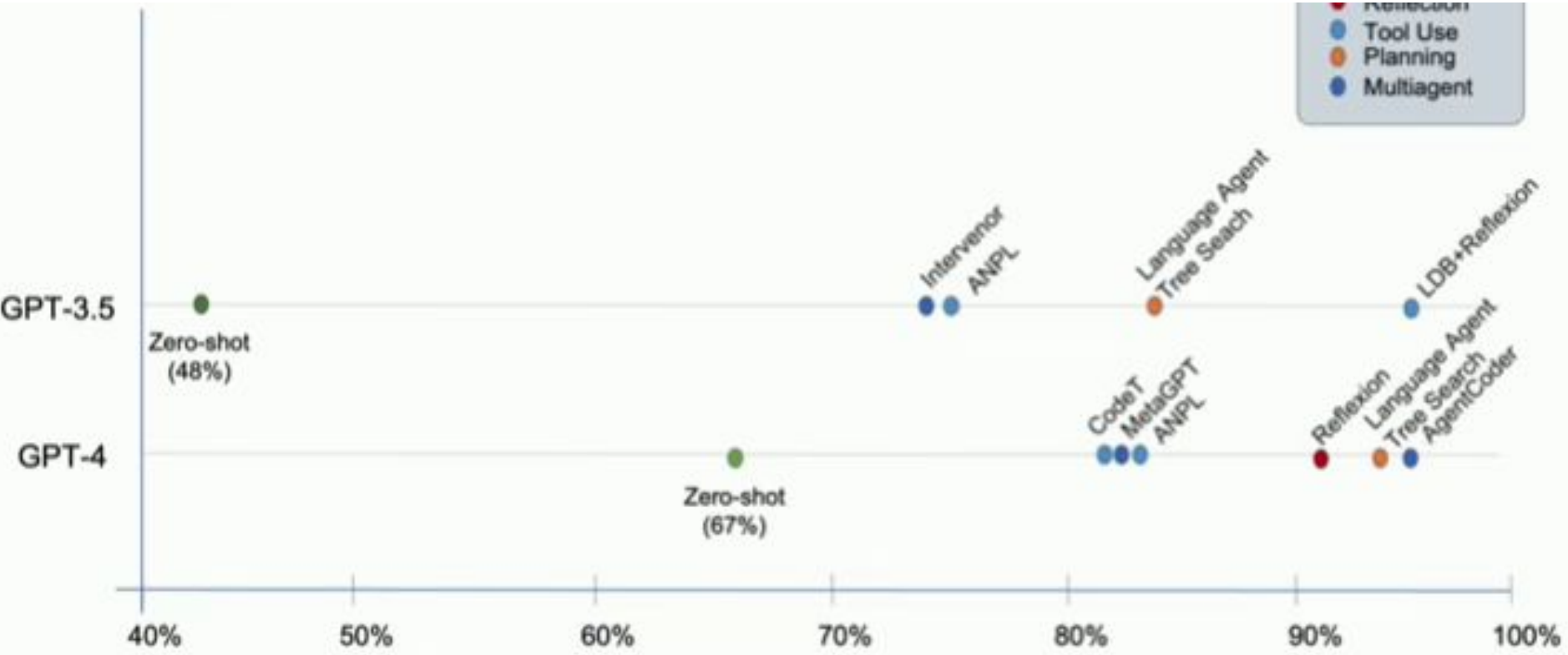
- *Write an essay outline on topic X*
- *Do you need any web research?*
- *Write a first draft.*
- *Consider what parts need revision or more research.*
- *Revise your draft.*



Models vs. Agents

Aspect	Models	Agents
Knowledge	Static, limited to training data	*Dynamic, extends via tools*
Functionality	Single inference or prediction	*Multi-step tasks with context retention*
Tools Integration	Not natively supported	*Built-in tool orchestration*
Logic Layer	Requires prompting	*Native cognitive reasoning frameworks*

Coding Benchmark (Human evaluation)



[Thanks to Joaquin Dominguez and John Santerre for help with analysis.]

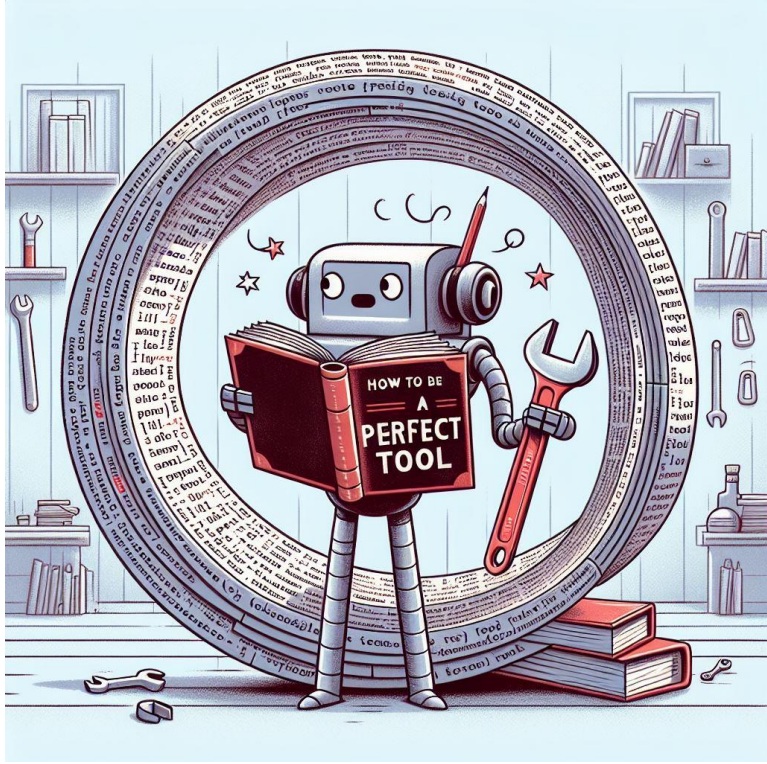
Andrew Ng

Cognitive Architecture of Agents

Core Components:

- **Model**: Decision-making through ***reasoning frameworks*** like ReAct, CoT, or ToT.
- **Tools**: APIs, data stores, or extensions for real-world interactions.
- **Orchestration Layer**: Manages memory, reasoning, and planning.

Process: Cyclical intake, reasoning, and action until goals are met.



Agents are nothing but just tools and text in a loop

Tools in Agents: **Enable agents to interact with external systems and data, overcoming model limitations**

Types of Tools:

- **Extensions**: Standardized bridges to APIs (e.g., Google Flights).
- **Functions**: Modular tasks executed client-side.
- **Data Stores**: Provide up-to-date information through vector embeddings.

Extensions: **Pre-built connections between agents and APIs, simplifying interaction.**

Features:

- Enable dynamic decision-making based on task needs.
- Include example configurations for scalability.

Example: Retrieve flight data using Google Flights API.

Functions:

Self-contained tasks managed client-side for flexibility and control.

Advantages:

- Handles security and timing constraints.
- Allows complex data transformations and offline execution.

Example: Calculate average sales per city in JSON format.

Data Stores: **Provide dynamic, real-time data access beyond static training knowledge.**

How It Works:

- Converts documents into vector embeddings for query matching.
- Supports formats like PDFs, spreadsheets, and website content.

Example: Retrieval-Augmented Generation (RAG) applications.

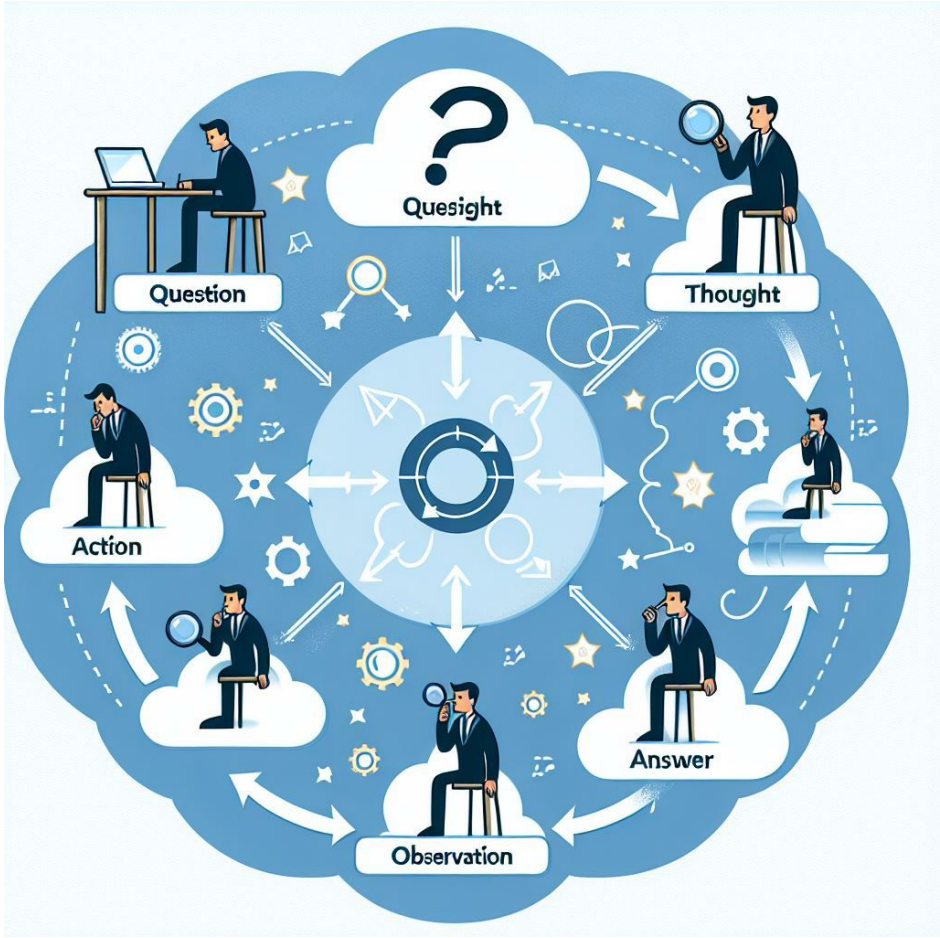
Orchestration Layer: ***Governs reasoning, decision-making, and task execution.***

Techniques:

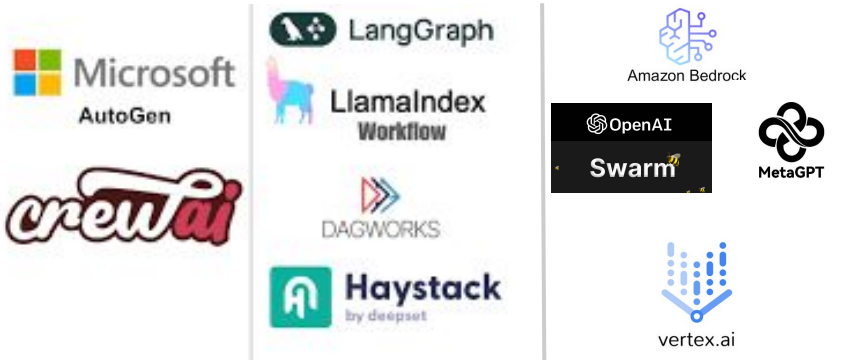
ReAct: Combines **reasoning and actions** iteratively.

Chain-of-Thought (CoT): Handles **multi-step reasoning** with sub-techniques like self-consistency.

Tree-of-Thoughts (ToT): Strategic exploration of **alternative solutions**.



Implementation Strategies

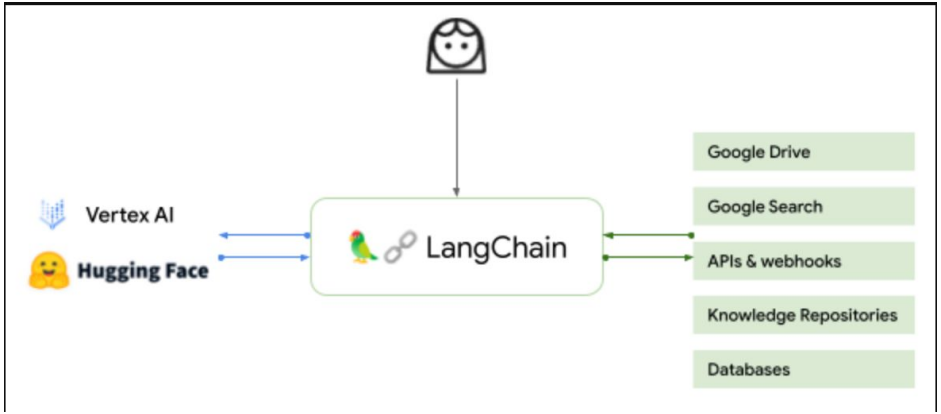


Quick Start with LangChain: **Combines reasoning, tool usage, and task orchestration.**

Production with Vertex AI: **Fully managed environment for agent building and refinement.**

Challenges:

- **Tool Selection**: Matching tools to task requirements.
- **Integration Complexity**: Managing multi-step workflows.
- **Performance**: Optimizing speed, cost, and accuracy for real-world deployment.



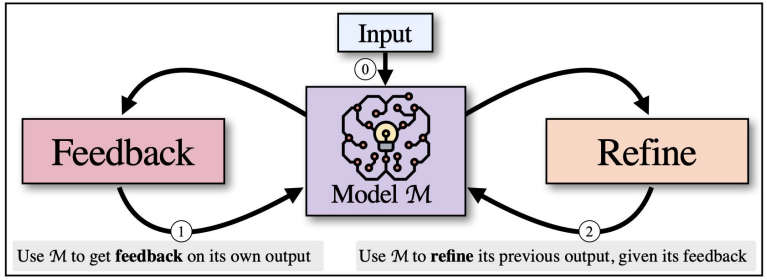
Agentic Design Patterns

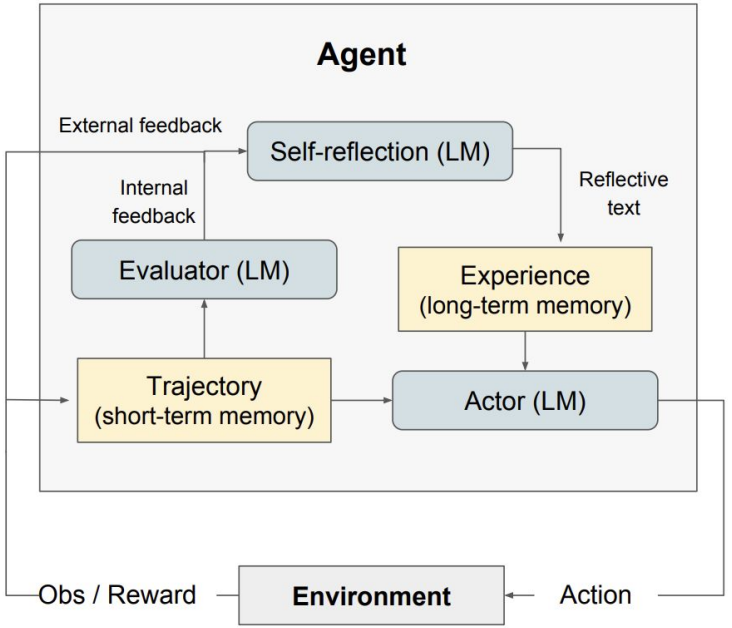
1. Reflection
2. Tool use
3. Planning
4. Multi-agent collaboration

Reflection

Verify & reflect the LLM output by external feedback (i.e. unit tests) & LLMs. Use the reflection to iterate the results.

- Self Refine
- Reflexion

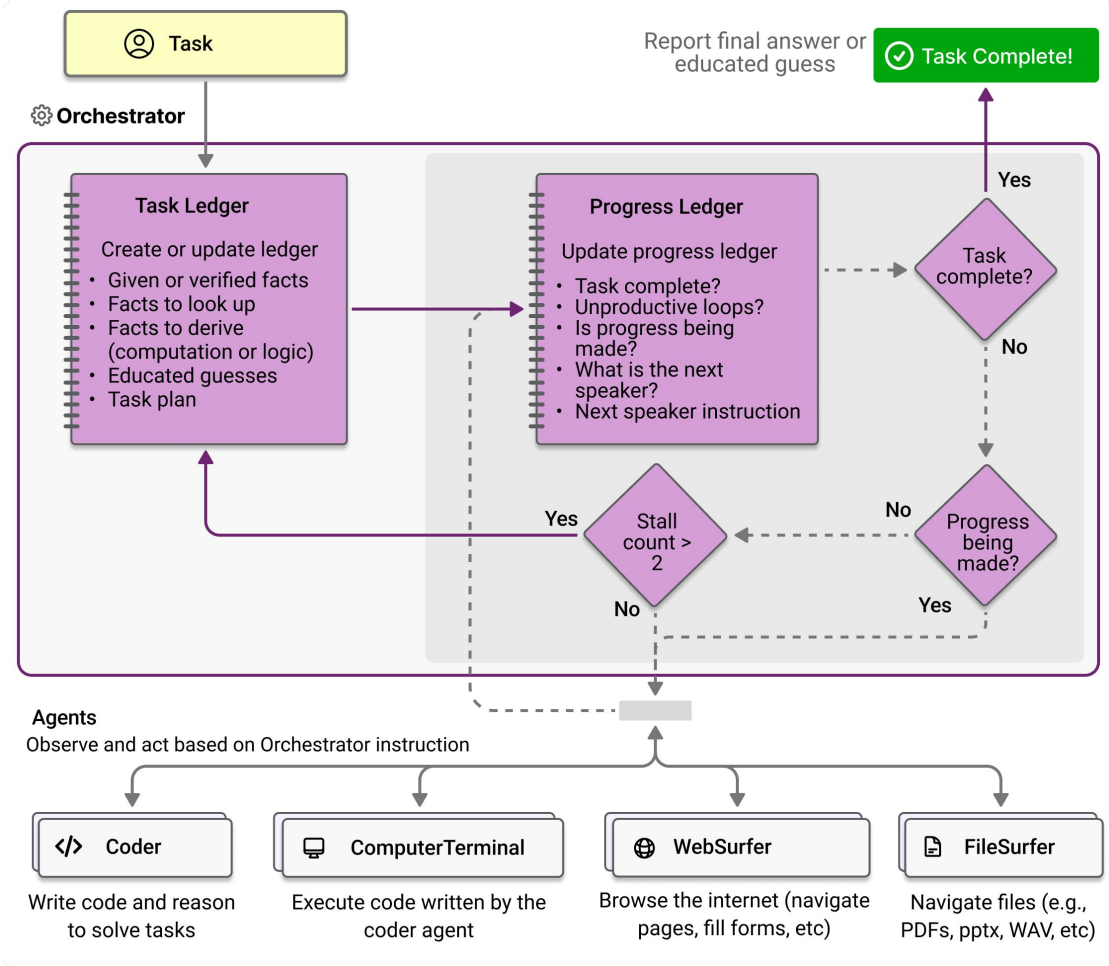




Self-Refine: Iterative Refinement with Self-Feedback. *Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, Peter Clark. NIPS 24.*

Planning

Plan tasks, track them while performing a task and reason over them if needed.

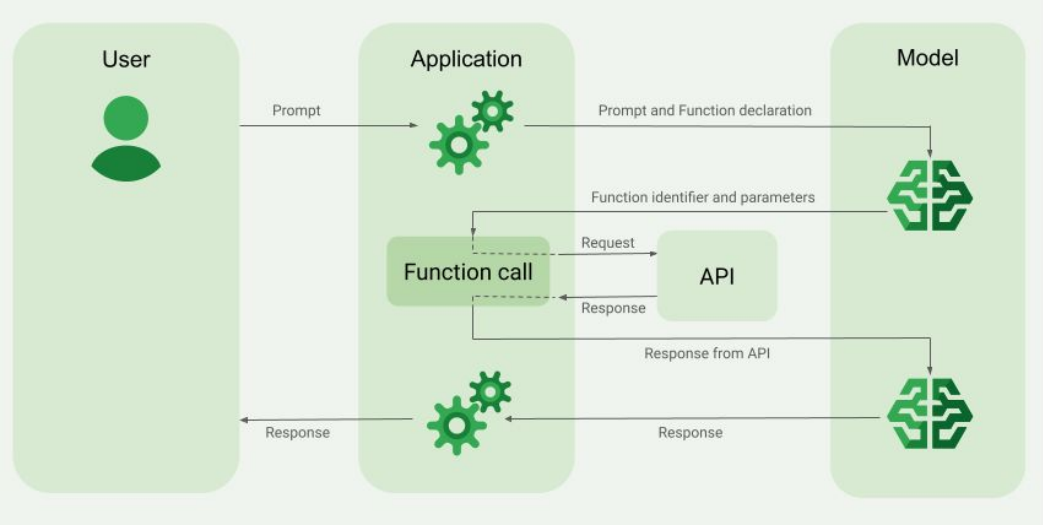


Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks. *Fourney et. al Microsoft Tech Report 2024.*

Tool Use

Connect with various tools & functions like: Browser, APIs, code exec, **custom code** (domain specific code hard for LLMs to generate), search engine etc.

Function Calling



Analysis
Code Execution
Wolfram Alpha
Binary Code Interpreter

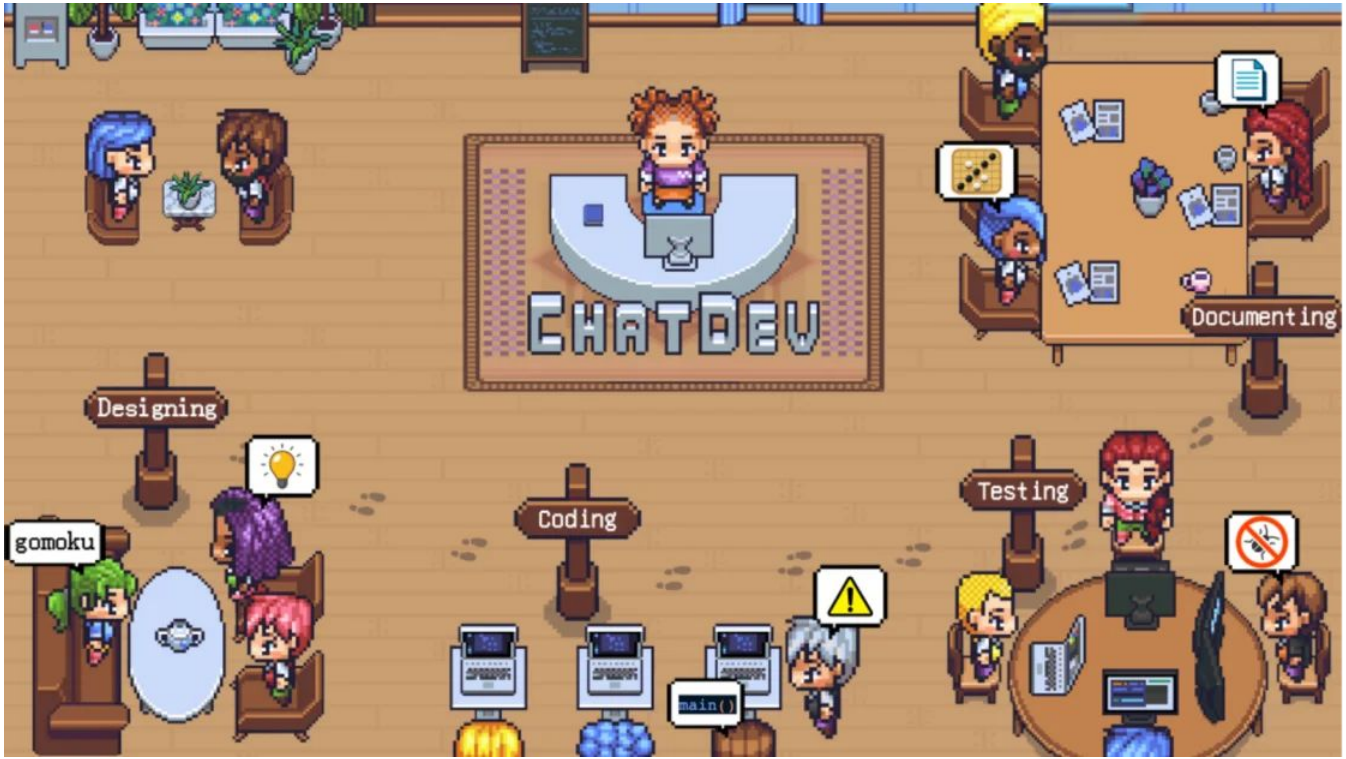
Research
Search engine
Web browsing
Wikipedia

Productivity
Email
Calendar
Cloud Storage

Images
Image generation (e.g., Dall-E)
Image captioning

Multi-agent Collaboration

Agents work together to solve a complex task.



ChatDev: Communicative Agents for Software Development. *Qian et. al. ACL 2024*

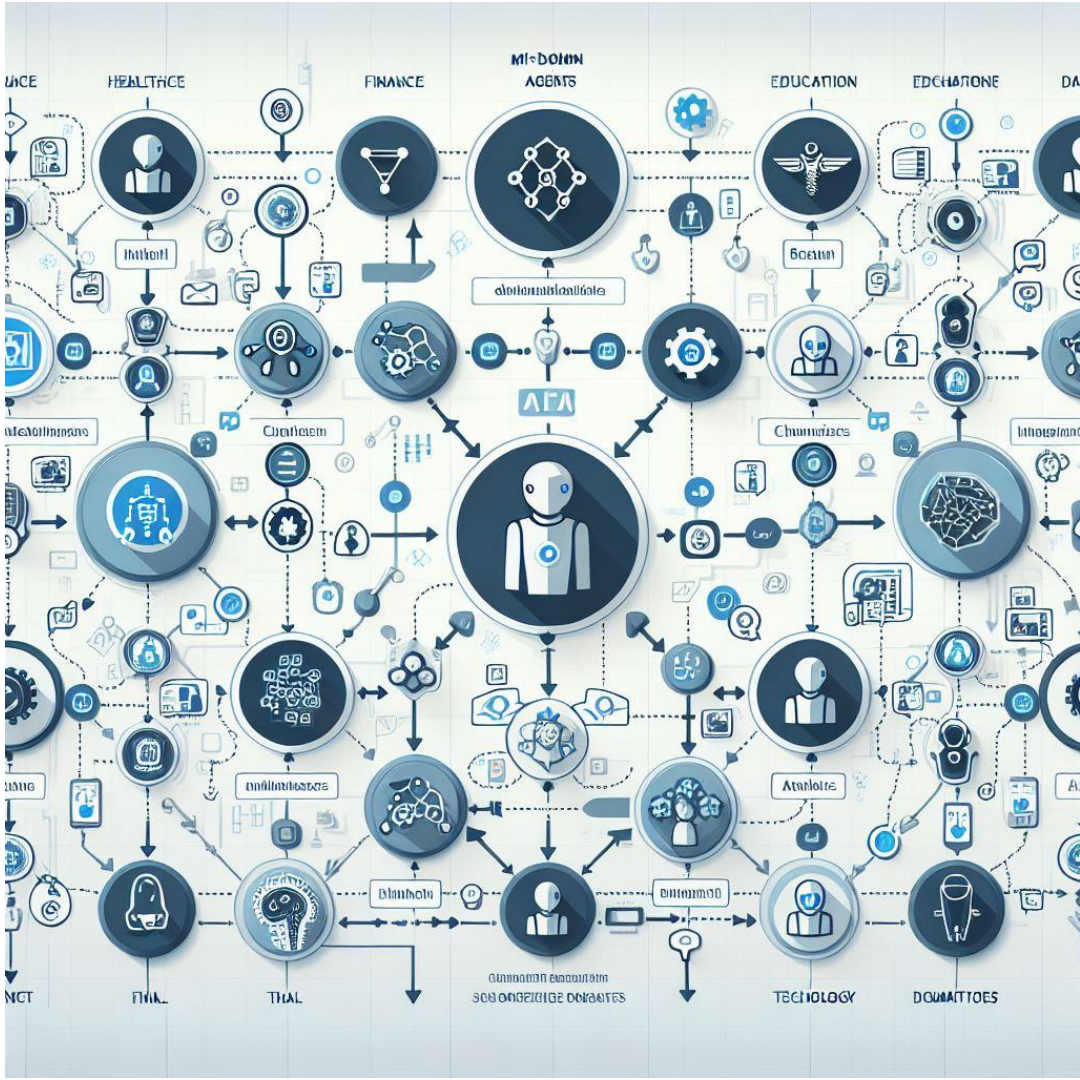
Future Directions:

Agent Chaining: Specialized *agents collaborating* for complex tasks.

Improved Reasoning: Advances in *cognitive frameworks like ToT*.

Scalability: Enhancing *tool and data integration* for broader applications.

Handling Complexity: Agent Computer Interface



Key messages:

Agents *combine reasoning, tools, and dynamic learning* for goal-oriented tasks.

Tools (Extensions, Functions, Data Stores) *bridge the gap between static knowledge and real-world interactivity*.

Implementing agents requires *robust orchestration and iterative refinement*.

AI Agents/Agentic LLMs is a *rapidly evolving* field.

Thank you