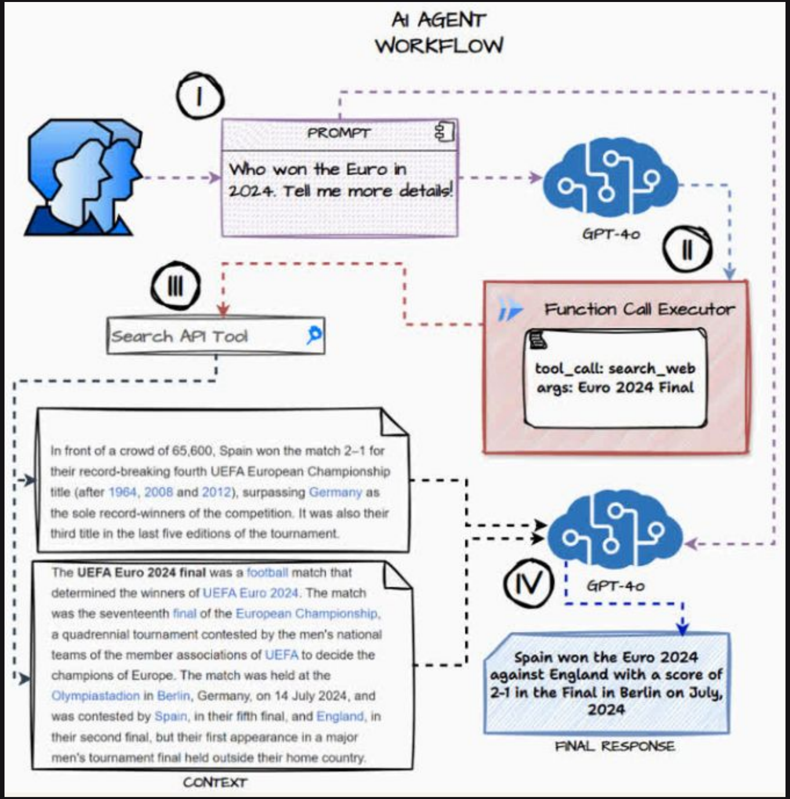
Analytics Vidhya

Best Practices to Build Effective Agentic AI Systems

Created & Narrated by:
Dipanjan Sarkar
Head of Community & Principal AI Scientist @ Analytics Vidhya
Google Developer Expert - ML & Cloud Champion Innovator
Published Author

Recap: What is an Agentic AI System



Who won the Euro in 2024. Tell me more details!

tool_call: search_web
args: Euro 2024 Final

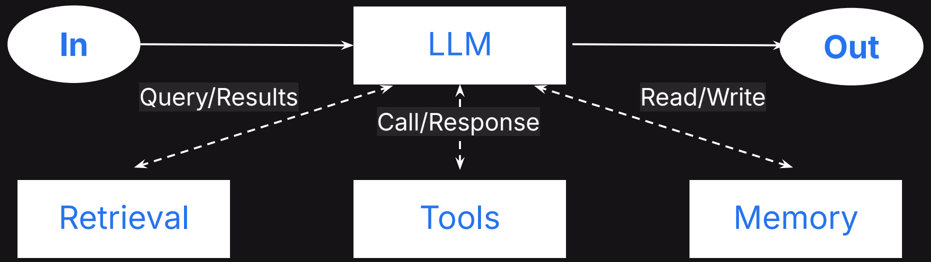
In front of a crowd of 65,600, Spain won the match 2–1 for their record-breaking fourth UEFA European Championship title (after 1964, 2008 and 2012), surpassing Germany as the sole record-winners of the competition. It was also their third title in the last five editions of the tournament.

The **UEFA Euro 2024 final** was a football match that determined the winners of UEFA Euro 2024. The match was the seventeenth final of the European Championship, a quadrennial tournament contested by the men's national teams of the member associations of UEFA to decide the champions of Europe. The match was held at the Olympiastadion in Berlin, Germany, on 14 July 2024, and was contested by Spain, in their fifth final, and England, in their second final, but their first appearance in a major men's tournament final held outside their home country.

Spain won the Euro 2024 against England with a score of 2-1 in the Final in Berlin on July, 2024

- An **Agentic AI System** is usually an autonomous system that operates independently over extended periods, using various tools and flows to accomplish complex tasks.
- Agentic AI Systems can be further categorized as:
  - **Workflows** are systems where LLMs and tools are orchestrated through predefined paths.
  - **Agents**, on the other hand, are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks.

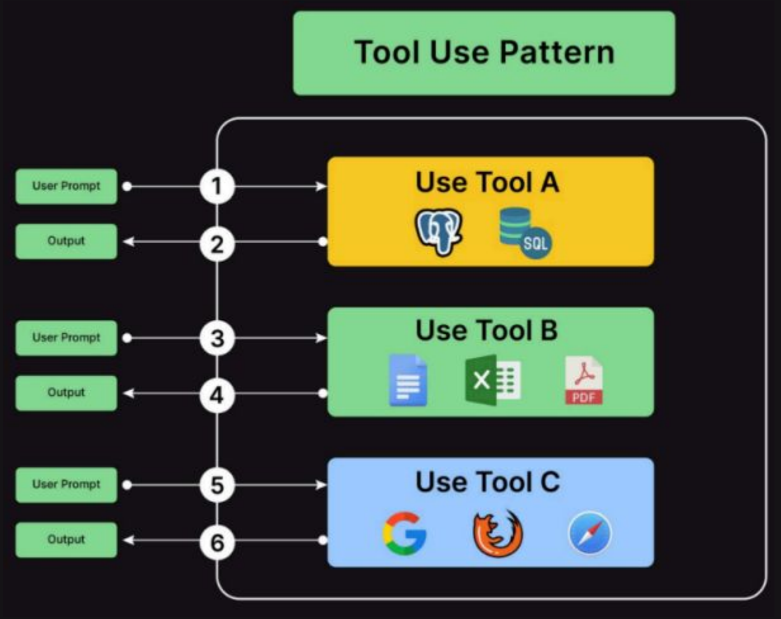
1. Build the Key Components for your Agent



- The basic building block of Agentic AI Systems is an **LLM enhanced with augmentations such as retrieval, tools, and memory**.
- Powerful LLM platforms have these in-built. When using APIs you would need to **connect the LLM with relevant tools, memory and databases** so that they can generate their own search queries, select appropriate tools, and determine what information to retain.
- Anthropic recommends focusing on **two** key aspects of the implementation:
  - Tailoring these capabilities to your specific use case.
  - Ensuring they provide an easy, well-documented interface for your LLM.

2. Start with Tool-Use Single ReAct Agents

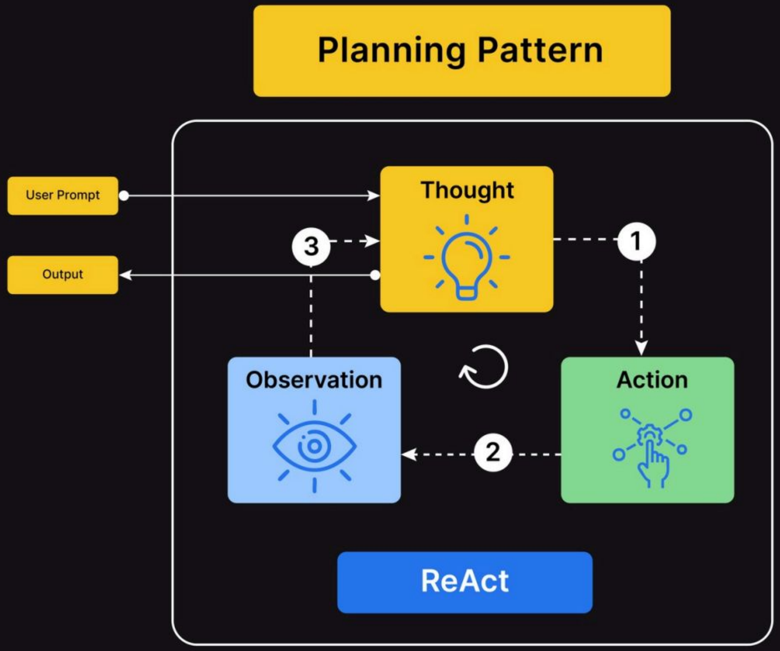
- Most **ReAct Tool-Use Agents** already have **planning capabilities**.
- These systems can easily handle **10-15 tools** easily.
- They can also handle multi-step and multi-tool call executions easily.
- Key drivers here include:
  - Well-defined tool schemas for accurate function calling.
  - Well-structured system prompt with detailed instructions.
  - Powerful LLMs already trained for function (tool) calling.
- Do NOT jump into multi-agent systems immediately - they are notoriously hard to control and debug regardless of present advancements.



Analytics Vidhya

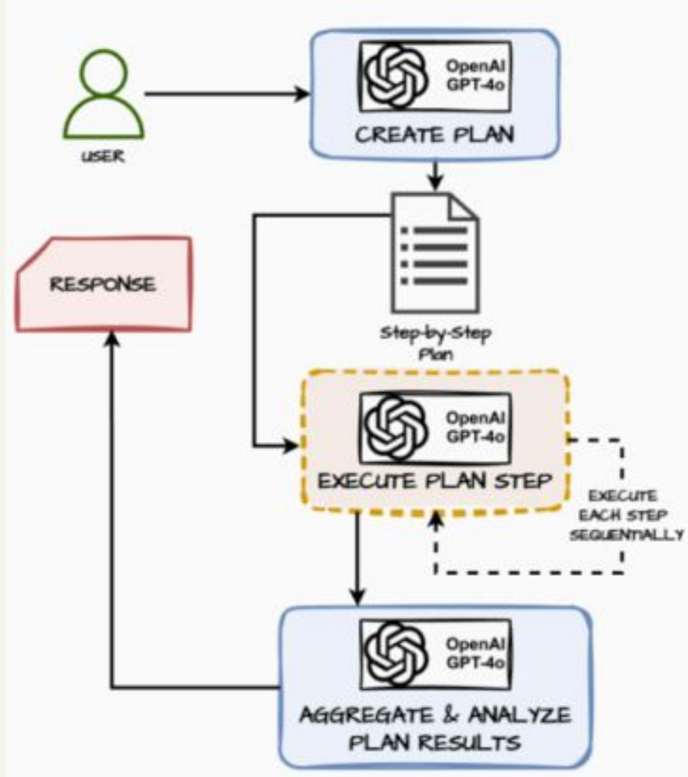
3. Planning Agents for Complex Task Execution

- Most **ReAct Agents** already have **planning built-in** so first start with simple ReAct Agents.
- If tasks are more complex and require explicit planning consider adding in additional planning modules in the Agent.
- Planning modules or patterns are typically:
  - **Static Planners** with Parallel Task Execution & Synthesis
  - **Dynamic Planners** with Task Execution, Reflection & Replanning
- Do **NOT** add extra planning steps or modules unless absolutely necessary as this **increases system latency**
- It is especially useful in complex reasoning, inference and validation scenarios.



Analytics Vidhya

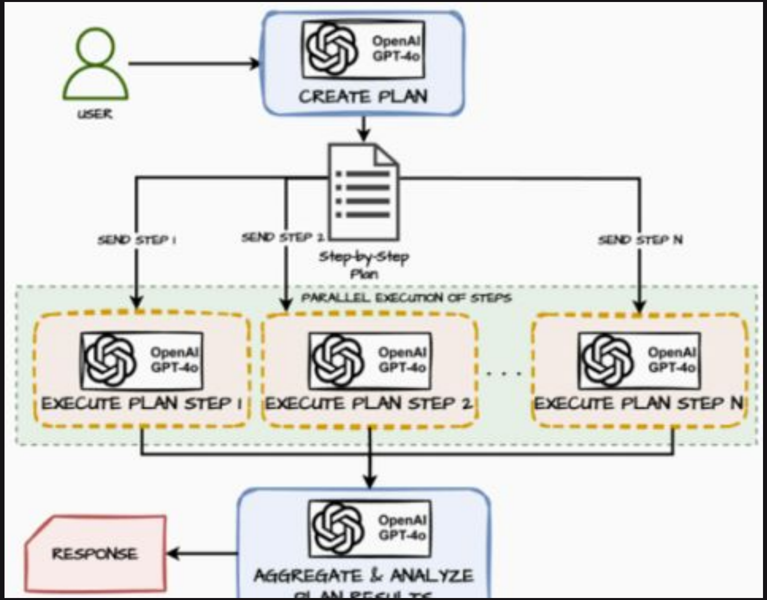
3. Planning Agents for Complex Task Execution



Dynamic Planners

- Use planning to break down a task into multiple steps.
- Executes one step at a time.
- Reflects on results of steps already executed.
- Uses reflection to replan remaining steps (if needed)
- Repeats till all steps are executed.
- Synthesizes results from all steps and generates final response.
- Useful when tasks may have dependencies among each other.

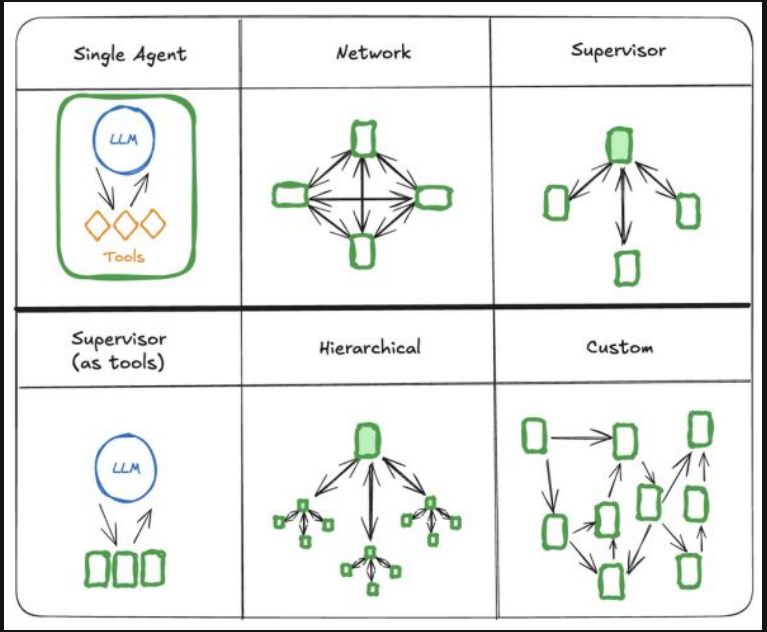
3. Planning Agents for Complex Task Execution



Static Planners

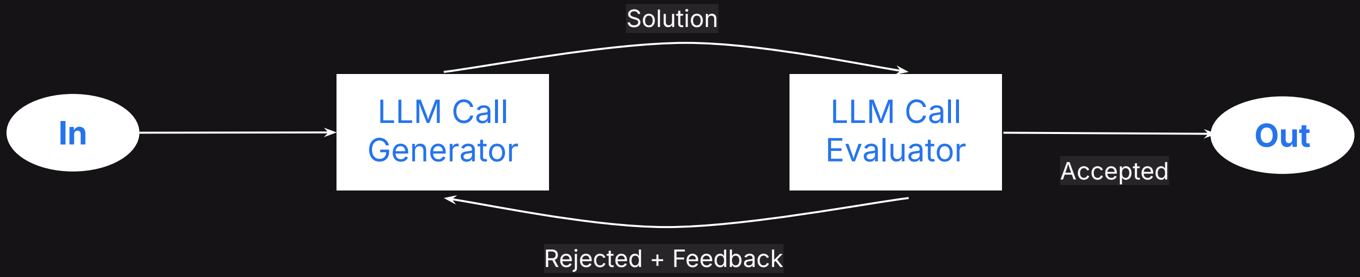
- Use planning to break down a task into multiple steps.
- Execute all steps in parallel
- Synthesize results from all steps and generate final response (map-reduce)
- Useful when steps do not have dependencies

4. Multi-Agent Systems for Efficient Task Distribution



- **Network:** Each agent can communicate with every other agent. Any agent can decide which other agent to call next
- **Supervisor:** Each agent communicates with a single supervisor agent. Supervisor agent makes decisions on which agent should be called next
- **Hierarchical:** Multi-agent system with a supervisor of supervisors. This is a generalization of the supervisor architecture and allows for more complex control flows
- Always start with simple supervisor or network architecture and then expand.
- Create separate agents based on specific processes, tasks and flows.

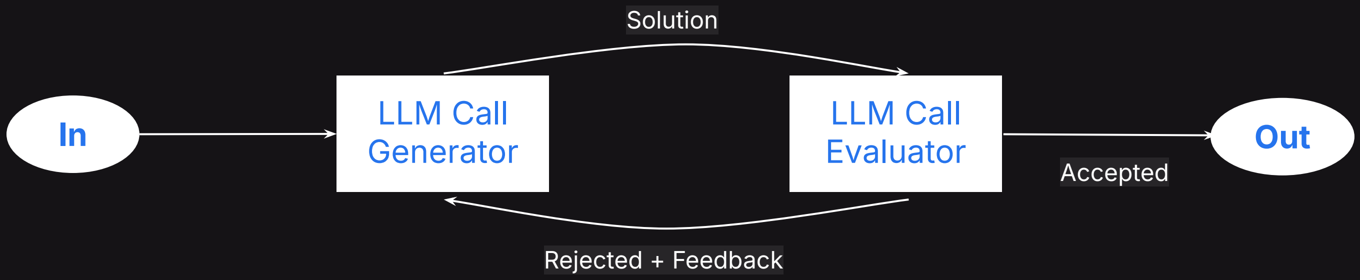
5. Reflection for Critiquing & Improvements



- Agents using reflection will leverage **one LLM call to generate a response** while **another provides evaluation and feedback in a loop**.
- This workflow is particularly effective when we have **clear evaluation criteria, and when iterative refinement provides measurable value**.
- **The two signs of good fit are:**
  - LLM responses can be demonstrably improved when a human articulates their feedback
  - The LLM can provide such feedback.

Analytics Vidhya

5. Reflection for Critiquing & Improvements



- This is analogous to the iterative writing process a human writer might go through when producing a polished document or when developed code is reviewed, tested and then improved.
- **Examples where reflection is useful:**
  - **Improving quality of RAG** retrieval and responses
  - **Judging, grading and critiquing** the quality of an LLM response
  - **Validating** specific criteria and guidelines e.g claims processing

Analytics Vidhya

Summary of Key Takeaways

1. Start by building key components of your Agentic AI System.
2. Start with Simple Tool-Use ReAct Single Agent Systems.
3. Add Planning only if the above fails and you need to add explicit planning modules for complex tasks.
4. Add Reflection for handling any tasks around grading, critiquing, improving responses at any step in the agent.
5. Add Routing to handle multiple flows or agents reliably.
6. Consider Multi-Agent systems when you have multiple processes, workflows, too many tools to handle and you can segregate tasks to specific agents with a set of tools.
7. Do not add in too many Agents unless absolutely necessary.
8. Monitoring and Debugging is super useful to check for common failure patterns.

Thanks!