Agentic Framework

Anup Ojah

Feb/03/2025

Understanding the Agentic Framework: A Blueprint for Autonomous Systems

In today's rapidly evolving world of artificial intelligence (AI) and machine learning (ML), the term "Agentic Framework" has emerged as a cornerstone for building intelligent, autonomous systems. But what does it mean, and why is it so significant?

For example, in the Manufacturing and Supply Chain industry, AI agents can optimize inventory management, predict maintenance needs for machinery, and dynamically adjust delivery routes based on real-time traffic and weather data.

The difference from existing solutions is that agentic solutions are based on Generative AI, which does not require pre-training of the model on specific data, and can reason based on that "knowledge" it already has. This makes agentic solutions extremely capable of handling a wide range of tasks without having to code them.

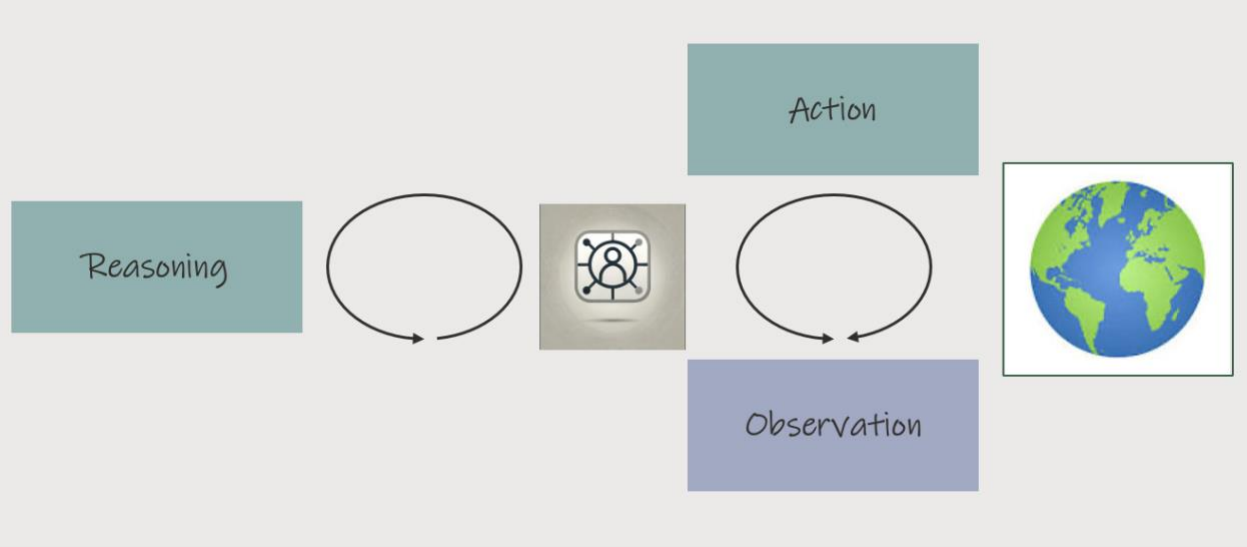
There are, however, a few drawbacks. One of them is that some use cases require specific domain knowledge, which foundation models are not familiar with. We will show how to solve this issue in this blog post by walking you through a wide range of design patterns to keep in mind before building an agent.

What is the Agentic Framework?

At its core, the Agentic Framework refers to a system design philosophy where AI agents are equipped with the ability to perceive their environment, make decisions, and take actions autonomously to achieve predefined goals. These agents operate independently, utilizing reasoning, planning, and learning capabilities.

ReAct – a new paradigm of agent that reasons and take actions. Below is a schematic diagram of how ReAct works.

Act: Let the model call specific tool
Observe: Pass the tool output back to the model
Reason: Let the model reason about the tool output to decide what to do next (e.g. call another tool or just respond directly)



There are four key principles of the agentic framework

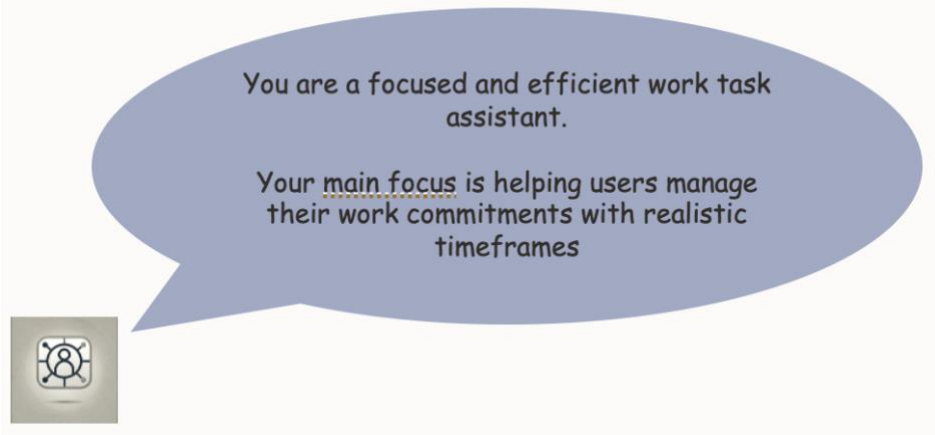
1. Autonomy: Agents must operate independently without requiring constant human intervention.
2. Goal-Orientation: Clear objectives guide the agent's behavior.
3. Adaptability: Agents should adjust their strategies based on real-time feedback.
4. Communication: Multi-agent systems rely on effective interaction between agents.

Core Components of the Agentic Framework

Here are six key components while building an Agentic Framework. As a developer, you need define the primary goal of the Agent and apply the following key components around it.

1. **Role Playing:**

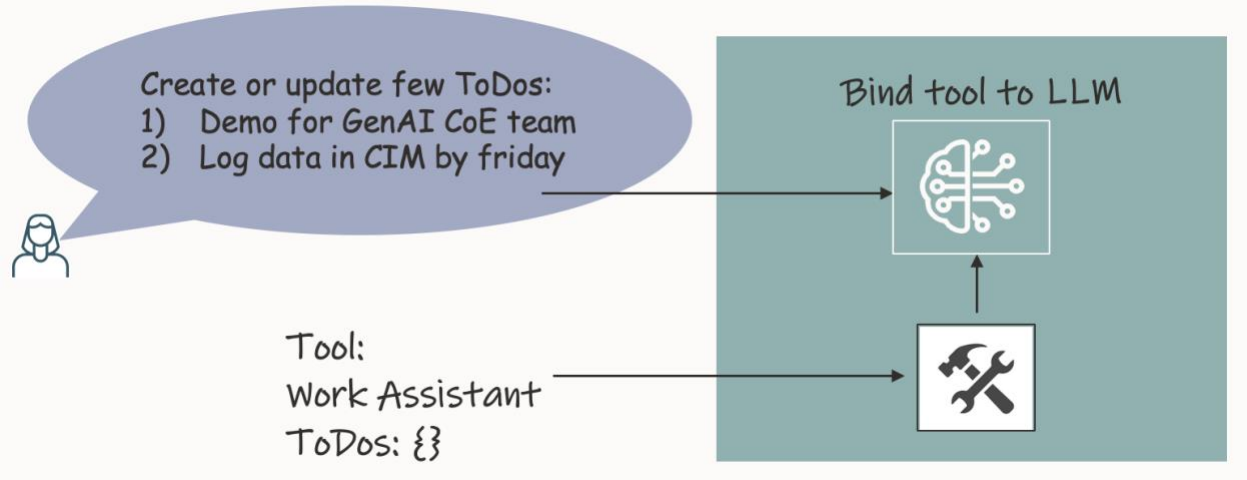
   Agents are assigned specific roles based on their expertise which ensures specialization and focus. This can be setup through system prompts that clearly define an agent's objectives.



2. **Tools:**

Agents leverage advanced tools, APIs, and frameworks to enhance performance. These enable them to perceive their environment, make decisions, and interact with other agents or systems effectively. For example:

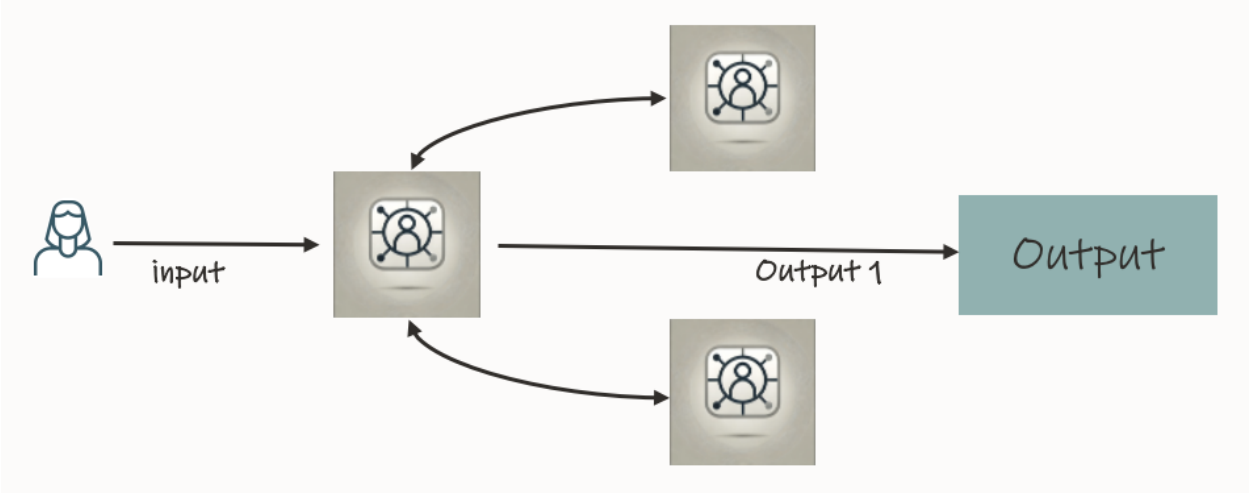
An intelligent work assistant agent that manages your daily Todo items. The agent can be provided a simple tool that keeps track of the user's list of tasks to be done along with calendars and status.



3. **Collaboration:**

Collaboration between agents ensures a synchronized effort towards achieving the overarching goal. Consider an Agent as a human that may not be able to solve a

business function on its own and may need to hand over responsibilities to a buddy (another Agent).

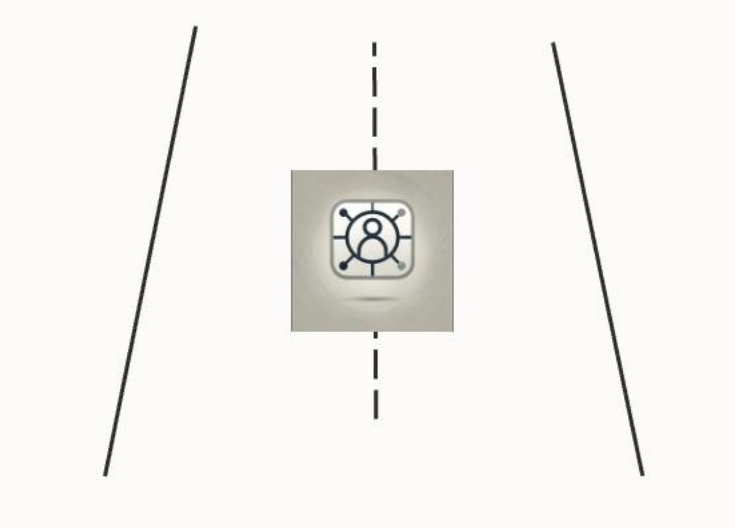


4. **Guardrails:**

In the context of the Agentic Framework, guardrails refer to the ethical, operational, and safety boundaries that ensure AI agents operate within predefined, responsible limits. These guardrails are designed to:

1. Ensure Ethical Compliance: Agents adhere to ethical principles, preventing unintended harm or bias.
2. Maintain Operational Control: Define clear boundaries for agent behavior to align with organizational goals.
3. Enhance Safety: Prevent actions that could lead to unintended or hazardous outcomes.

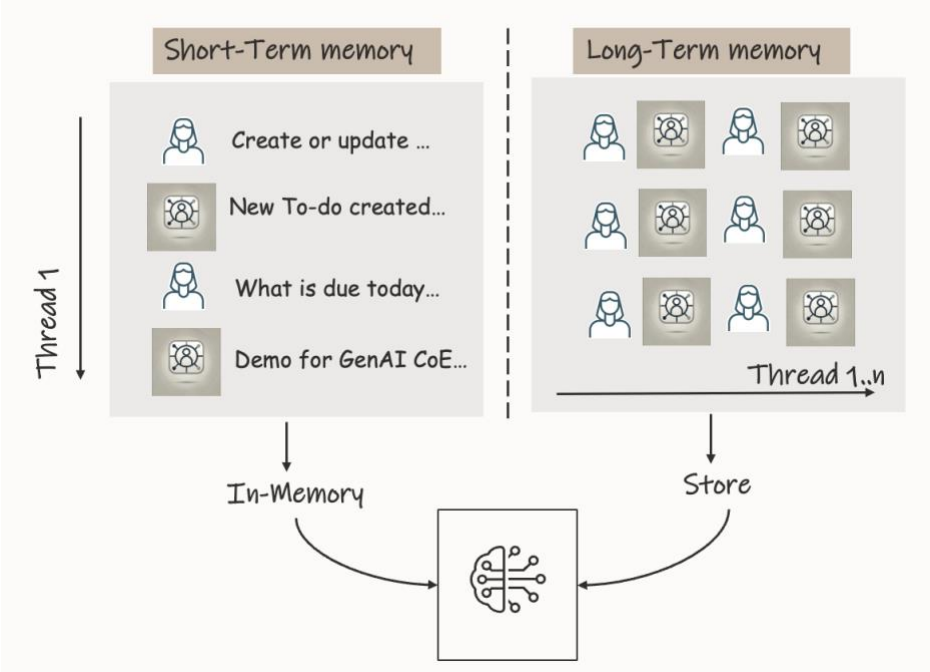
Guardrails act as a critical safeguard in multi-agent systems, providing a structure to guide agents' autonomous actions while ensuring accountability and reliability.



5. **Memory:**

Agents are equipped with memory systems to store context, learn from past interactions, and improve future decisions. There are two type of memory systems, short-term and long-term.

Example, using Short-term (within-thread) memory a Chatbot can persist conversational history and / or allow interruptions in a chat session. Long-term (cross-thread) memory a Chatbot can remember information about a specific user across all chat sessions.



6. **Entity Memory:**

Entity extraction is a crucial step in storing memory in an Agentic Framework as it enables the AI agents to identify and organize meaningful information from vast data inputs. Here's why it is significant:

Entity extraction organizes key information (e.g. names, dates, terms) to streamline context management, enabling efficient memory storage and retrieval. It enhances recall by linking extracted entities to structured memory, enriching knowledge graphs for reasoning and pattern recognition. Personalized interactions are achieved by remembering user-specific details, while scalability ensures standardized data sharing across multi-agent systems. Extracted insights also improve decision-making, helping agents deliver tailored solutions, detect anomalies, and adapt to complex environments effectively.

```
================================ Human Message =================================

My name is Anup. I live in Louisville with my wife and my 2 kids. I work for Oracle corporation
================================== Ai Message ==================================
Tool Calls:
  UpdateMemory (call_cJbeevowQKQYWoIgDPuvMqZ3)
 Call ID: call_cJbeevowQKQYWoIgDPuvMqZ3
  Args:
    update_type: user
================================= Tool Message =================================

updated profile
================================== Ai Message ==================================

Thanks for the update, Anup! How can I assist you today?
```

Store in Memory

```
{'name': 'Anup', 'location': 'Louisville', 'job': 'Oracle corporation', 'connections': ['wife', '2 kids'], 'i
nterests': []}
```

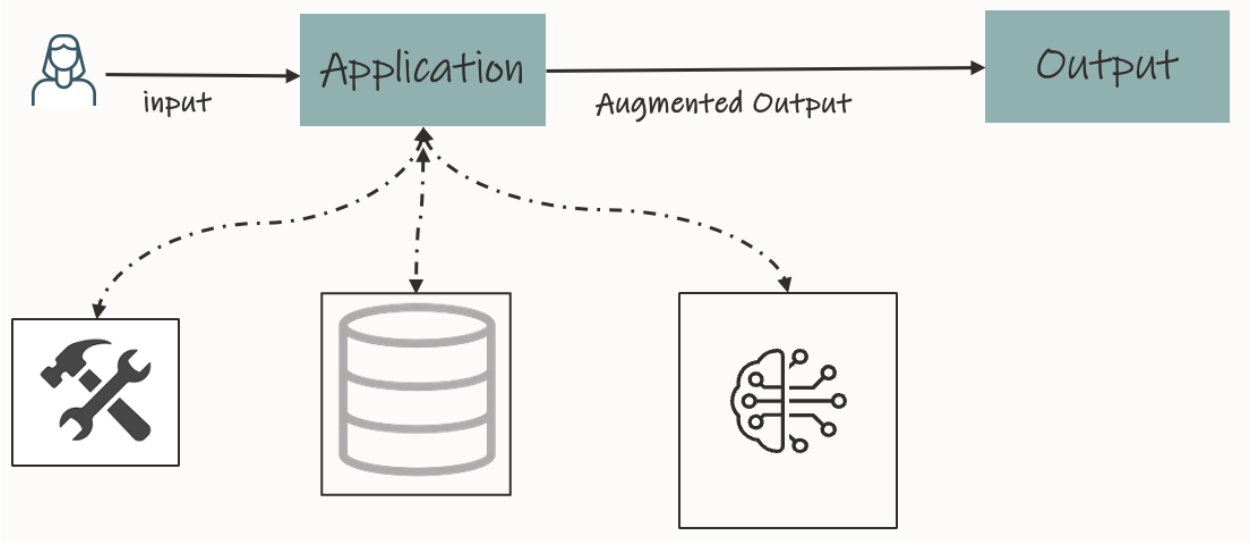
Design patterns using Agentic Framework

1. **Retrieval Augmented Generation (RAG):**

   This design pattern integrates external knowledge retrieval to augment the performance of language models. Agents can retrieve and incorporate data dynamically, improving accuracy and relevance.

   RAG enhances foundational models (FMs) by addressing domain expertise gaps and minimizing hallucinations. FMs, trained on general datasets, often lack specialized knowledge. RAG resolves this by dynamically retrieving relevant, domain-specific information during inference. For instance, in healthcare, RAG can access trusted medical databases to provide accurate, context-aware responses.

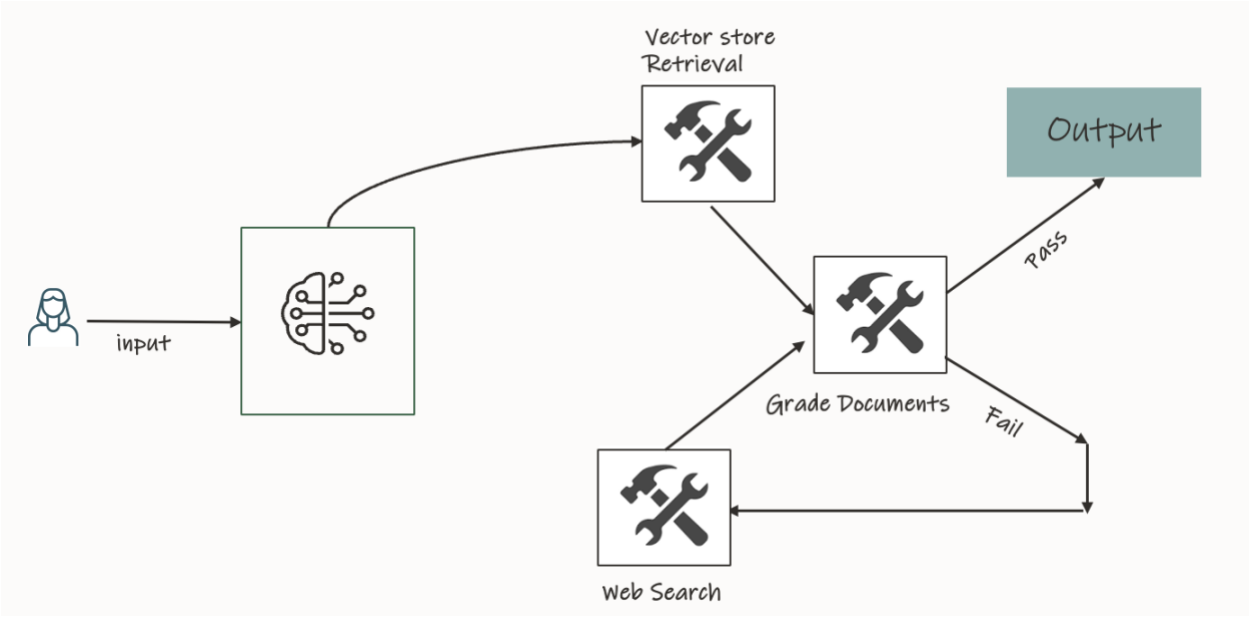
   To combat hallucinations, RAG grounds outputs in factual evidence from external knowledge sources. Retrieved documents are integrated with the model's generative process, ensuring responses are based on verifiable data. By ranking and filtering sources, RAG prioritizes high-quality, reliable information, reducing the risk of unsupported claims. This approach also allows for transparency, as retrieved data can be cited within the response.



2. Corrective RAG (CRAG)

Corrective Retrieval-Augmented Generation (CRAG) is an advanced extension of RAG that incorporates iterative feedback and correction loops to refine AI-generated outputs. While traditional RAG enhances foundational models (FMs) by retrieving external knowledge, CRAG goes further by introducing a self-correcting mechanism to mitigate errors, hallucinations, and misinformation. For Example:

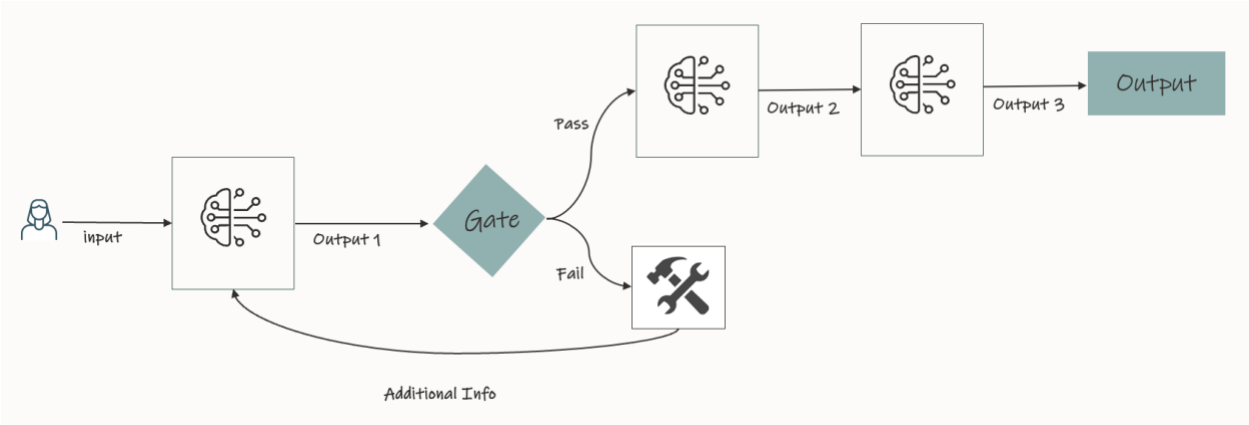
In Healthcare, ensure that AI generated medical advice aligns with trusted clinical guidelines.



3. **Chaining:**

Chaining breaks down complex tasks into manageable steps or chains where the output of one step serves as the input to the next. This design enhances modularity, control, and scalability when building intelligent systems. This pattern also allows to introduce branching logic based on the LLM's output or intermediate results. For example:

- A sentiment analysis task could trigger a sentiment mitigation process if the result is negative.
- Intermediate steps might interact with APIs or databases for data validation.



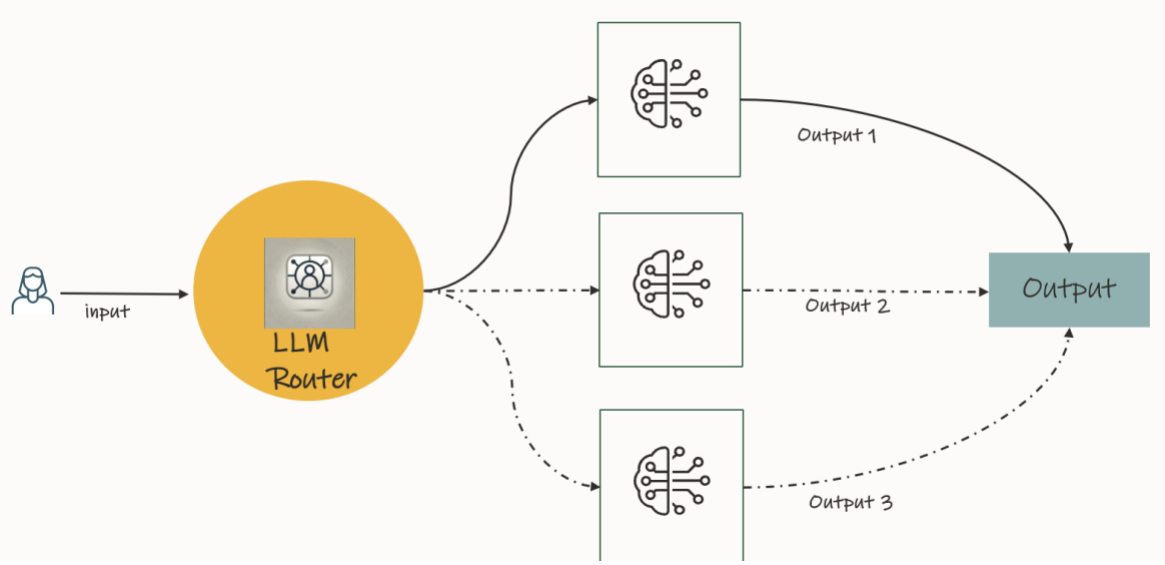
4. **Routing**:

An Agentic Framework Routing Pattern refers to the design strategy used to direct tasks, inputs, or queries to the appropriate agents or workflows within an Agentic Framework. This pattern ensures that tasks are handled by the most suitable agent or sequence of agents, optimizing for efficiency, accuracy, and resource utilization.

3.1 **Dynamic Routing:**

Routing decisions are often dynamic, based on real-time analysis of input data, task requirements, or resource availability. For Example:

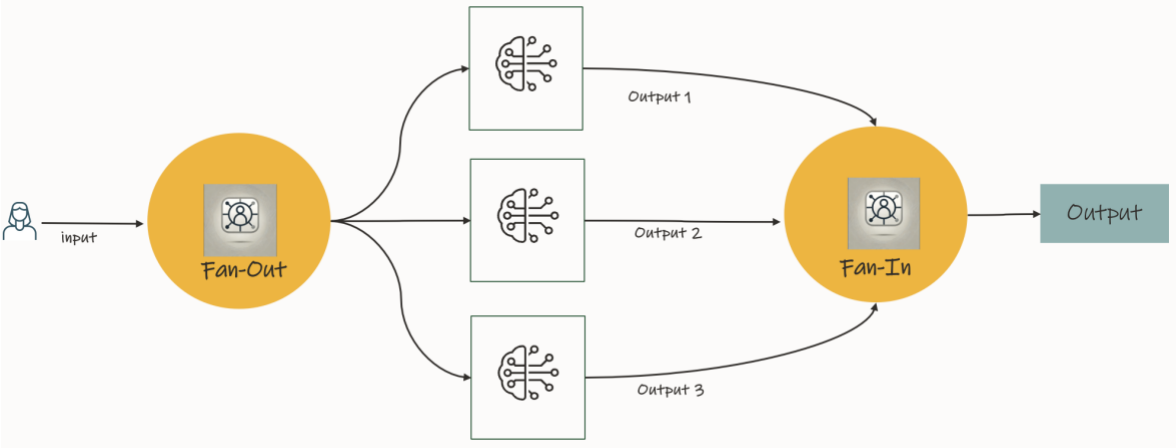
A prompt involving natural language query against a system of record might be routed to a specialized NL2SQL agent instead of a general-purpose LLM.



3.2 **Parallel Routing:**

Parallel Routing in an Agentic Framework refers to the process of distributing tasks across multiple agents or nodes simultaneously (fan-out) and then aggregating or consolidating their outputs (fan-in) to produce a final output. This pattern is particularly useful for improving efficiency and handling complex workflows where subtasks can be processed independently. For Example:

A long document is split into paragraphs. Each paragraph is sent to a separate summarization agent (fan-out). Agents summarize their respective paragraphs simultaneously (parallel processing). Individual summaries are later merged into a cohesive summary for the entire document (fan-in).

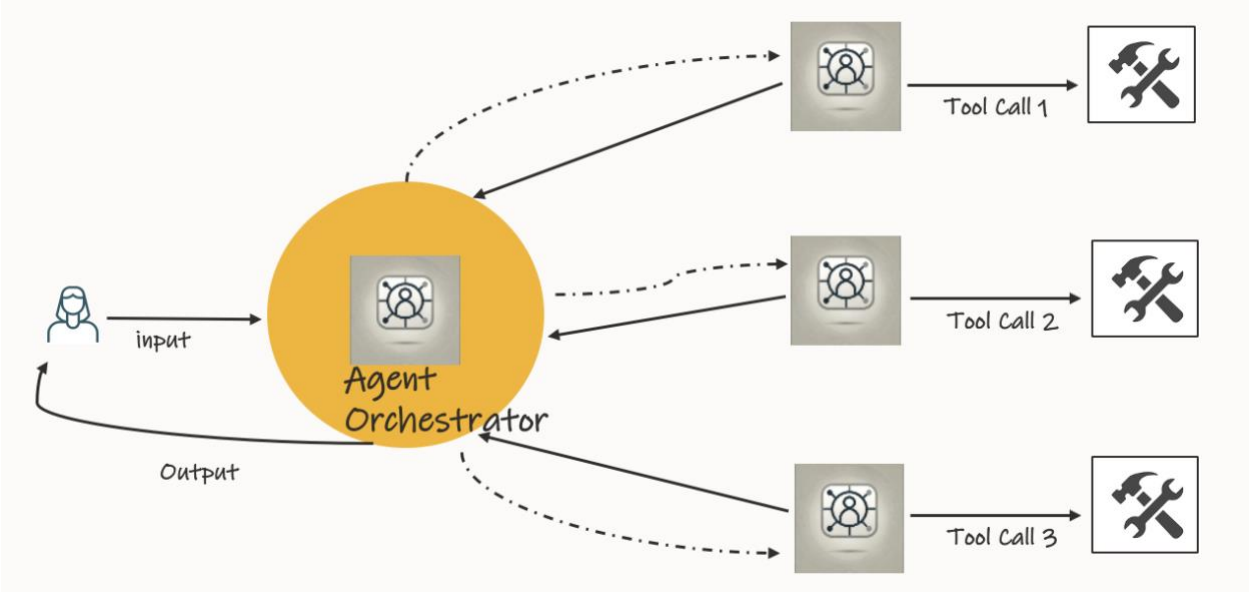


5. **Orchestration:**

An Orchestration Pattern in an Agentic Framework refers to the structured coordination of agents and processes to execute a task or workflow effectively. Here an LLM with Reasoning capabilities can be used as the Orchestrator, often called a Master agent. The Master agent dynamically assigns (unlike Routing pattern, the subtasks aren't pre-defined) task to worker agents, responsible for coordinating and overseeing the collaborative tasks of worker agents, and later reduces the final output.

This pattern is well suited for building complex Multi-Agent System. For Example:

A user uploads a pdf that has both text and images embedded to a chatbot and ask to summarize. The Master agent decides to orchestrate multiple worker agents – one for analyzing text to text, and another one to analyze image to text. The final output is later synthesized (post-process) and presented to the user.



6. **Human-in-the-Loop (HITL):**

The Human-in-the-Loop (HITL) pattern in an Agentic Framework integrates human oversight and intervention into automated AI-driven workflows. This pattern is essential for ensuring accuracy, safety, and ethical decision-making in scenarios where AI alone may be insufficient or untrustworthy.

How HITL Works:

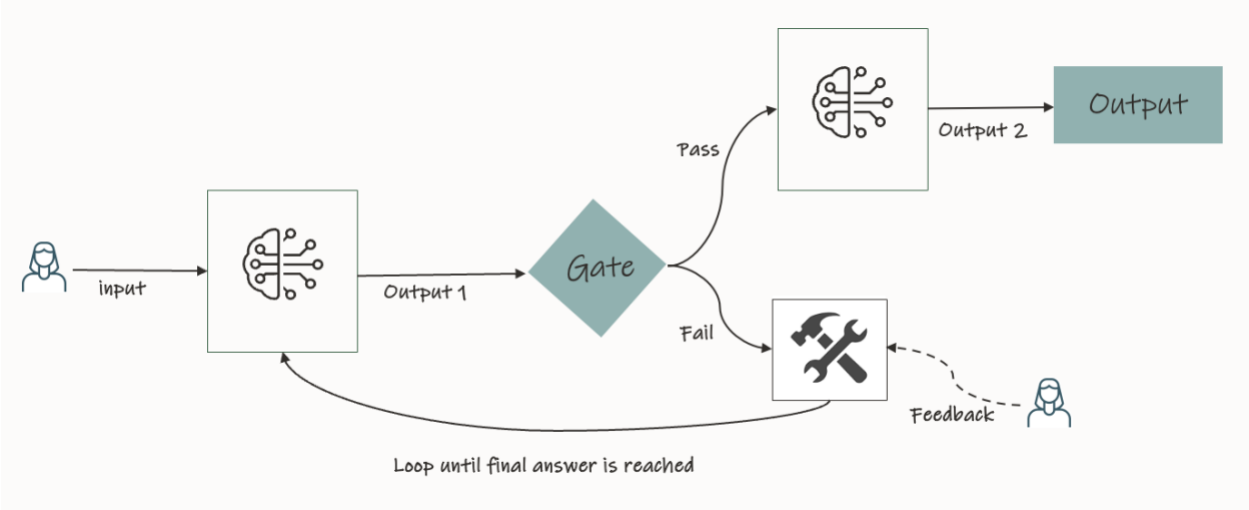
Validation & Oversight – Humans review AI-generated decisions before execution.

Intervention & Correction – Users can modify or override AI actions to prevent errors.

Continuous Learning – AI learns from human feedback, improving over time.

Escalation Mechanism – AI escalates uncertain or conflicting cases to human experts.

By integrating human judgment, HITL enhances reliability, reduces bias, and improves adaptability, making AI safer and more accountable in critical applications.



Choosing the right Agentic Framework

Choosing the right Agentic Framework to build Agents is crucial for ensuring that your system is efficient, scalable, and aligned with your goals. The strategy involves

evaluating your project requirements, understanding the capabilities of various frameworks, and ensuring they meet your technical and operational needs.

There exist many frameworks such as LlamaIndex, Crew.AI, AutoGen, Open AI Swarm, LangGraph, etc., that can help you start building production grade Agents. But the key to choose one lies in adopting a framework that is simple to use and offers minimum abstraction layer (tight integration) from the core LLM of your choice.

Here at Oracle, we offer a fully managed LLM service, called OCI Generative AI that provides access to Cohere and the Meta Llama models. Oracle also offers bring-your-own LLM (from Hugging Face or any third party) through OCI Data Science Quick Actions. Both the OCI GenAI Service and OCI Data Science Quick Actions has tight integration with the LangChain community and the LangChain framework is constantly updated as changes or new functionalities get added to the underlying LLM services.

As an Oracle AI developer, I decided to adopt a framework that supports LangChain as an underlying abstraction layer. Based on my experimentation (simple to use and offers minimum abstraction layer), I decided to proceed with LangGraph. LangGraph is a cutting-edge Agentic Framework designed for building and managing intelligent agents that perform complex, multi-step workflows. It emphasizes modularity, collaboration, and scalability, allowing developers to construct agents capable of handling diverse tasks in a coordinated and efficient manner. If your project involves complex, dynamic workflows or requires multimodal processing, LangGraph is an excellent choice. If on the other hand you have a project that needs basic routing or chain-based agentic pattern, LangChain instead of LangGraph (to avoid additional abstraction layer) should be considered. Other frameworks like Swarm or LlamaIndex are also great options if the need arises due to business or customer requirements.

Challenges and Future Directions

While the Agentic Framework holds promise, challenges such as ethical considerations, accountability, and transparency must be addressed. Researchers are actively exploring ways to make these systems more interpretable and aligned with human values.

Conclusion

The Agentic Framework is revolutionizing how we design intelligent systems by enabling autonomous agents to reason, adapt, and collaborate effectively. Its applications span industries, from optimizing supply chains to advancing healthcare, demonstrating its transformative potential.

While challenges like ethical considerations and transparency remain, frameworks like LangGraph and LangChain offer robust solutions to build scalable, efficient systems. These innovations ensure AI agents operate autonomously while maintaining accountability and alignment with human values.

As this framework evolves, it promises to reshape industries and solve real-world challenges more effectively.

In my next blog, I will be sharing code to create Agents based on LangGraph. Stay tuned …

For more information on OCI Generative AI and OCI Data Science, visit the following resources:

- OCI Data Science
- OCI Generative AI Service
- OCI Generative AI Agents
- What are AI Agents?
- Why generative AI with Oracle?
- What is retrieval-augmented generation (RAG)?