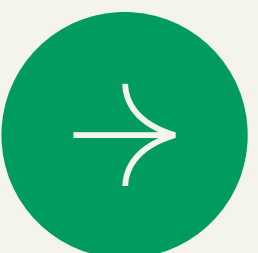
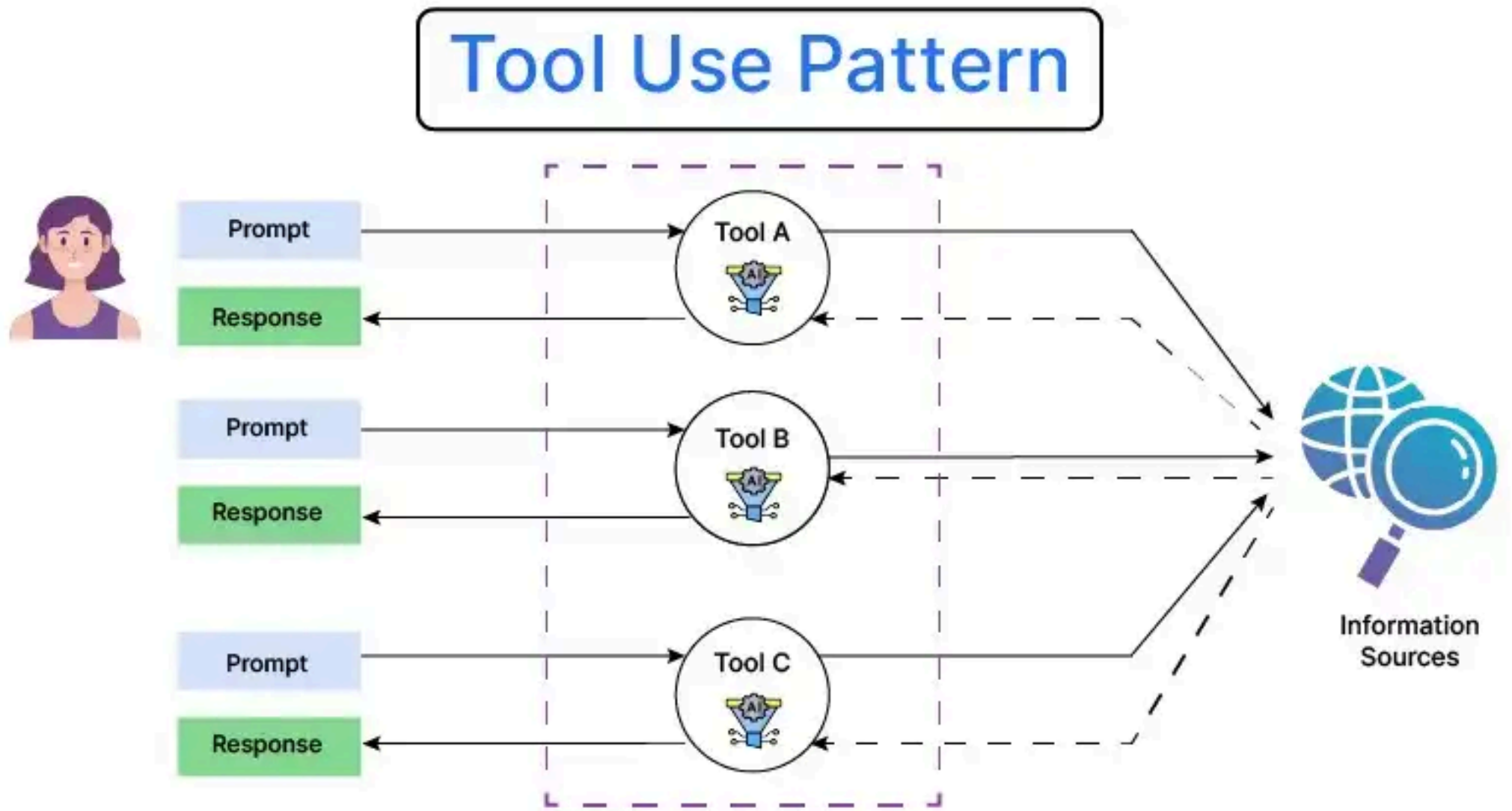


Guide to Agentic AI Tool Use Pattern



Why Tool Use Agents?

Tool Use Pattern in Agentic AI

Web Search Tool

🔍 Top 5 Attractions in Delhi

P give me 5 places to visit in Delhi

Here are 5 notable places to visit in Delhi:

1. Red Fort
2. Qutub Minar
3. India Gate
4. Humayun's Tomb
5. Lotus Temple

Would you like more details about any of these places or suggestions for additional sites to see in Delhi?

Copy Ret


 Claude

Claude can make mistakes. Please double-check

Code Execution Tool

Thought for a couple of seconds

Certainly! Merge Sort is a classic divide-and-conquer algorithm that efficiently sorts a list by recursively dividing it into smaller sublists, sorting those sublists, and then merging them back together. Below is a Python implementation of the Merge Sort algorithm, complete with comments and an example of how to use it.

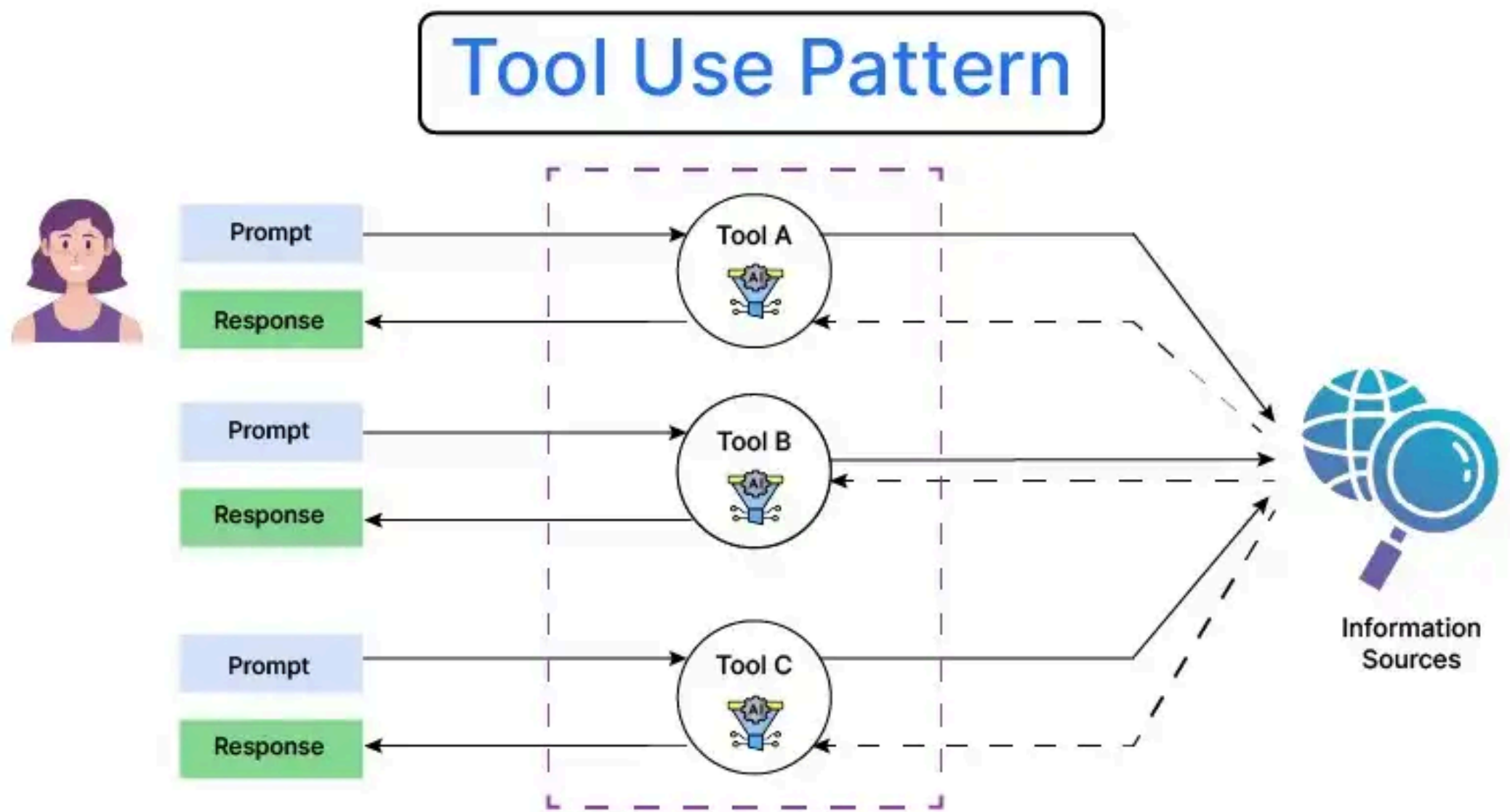
python  Copy code

```
def merge_sort(arr):  
    """  
    Sorts an array in ascending order using the Merge Sort algorithm.  
  
    Parameters:  
    arr (list): The list of elements to be sorted.  
  
    Returns:  
    list: A new sorted list.  
    """  
    # Base case: a list of zero or one elements is already sorted  
    if len(arr) <= 1:  
        return arr  
  
    # Recursive case: split the list into two halves  
    mid = len(arr) // 2  
    left_half = merge_sort(arr[:mid])  
    right_half = merge_sort(arr[mid:])  
  
    # Merge the sorted halves
```

GPT-4o mini

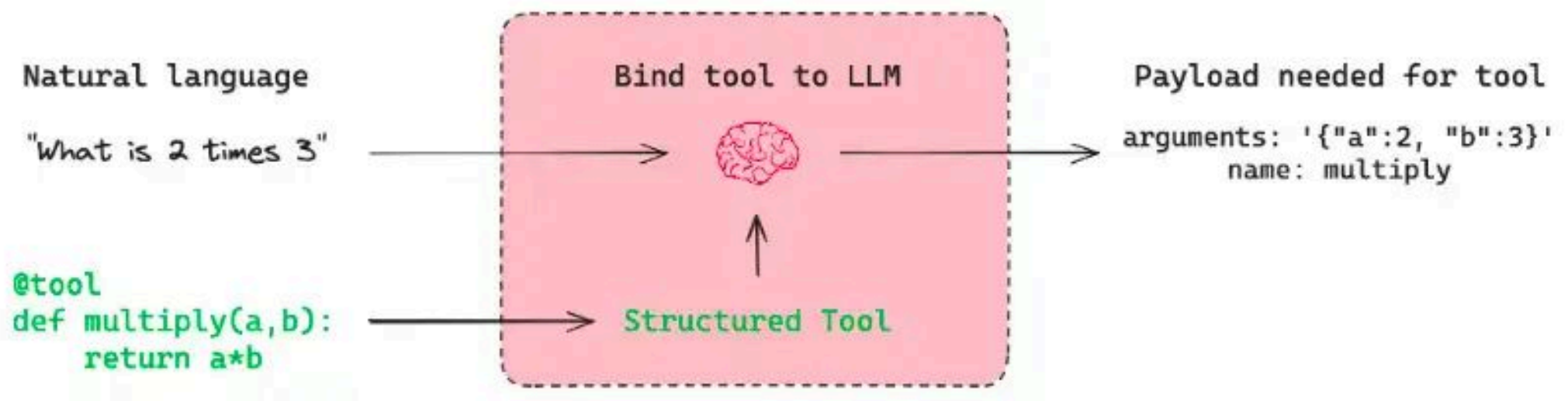
- **Tool Use Pattern in Agentic AI enables language models to interact with external systems and APIs**
- **Addresses the traditional constraints of LLMs, which are often limited to outdated pre-trained data**
- **These Agentic AI systems can autonomously select, utilize, and coordinate multiple tools to perform complex tasks without constant human input**
- **Examples include agents that conduct real-time web searches, perform sentiment analysis, and more**

Tool Use Pattern – Architecture



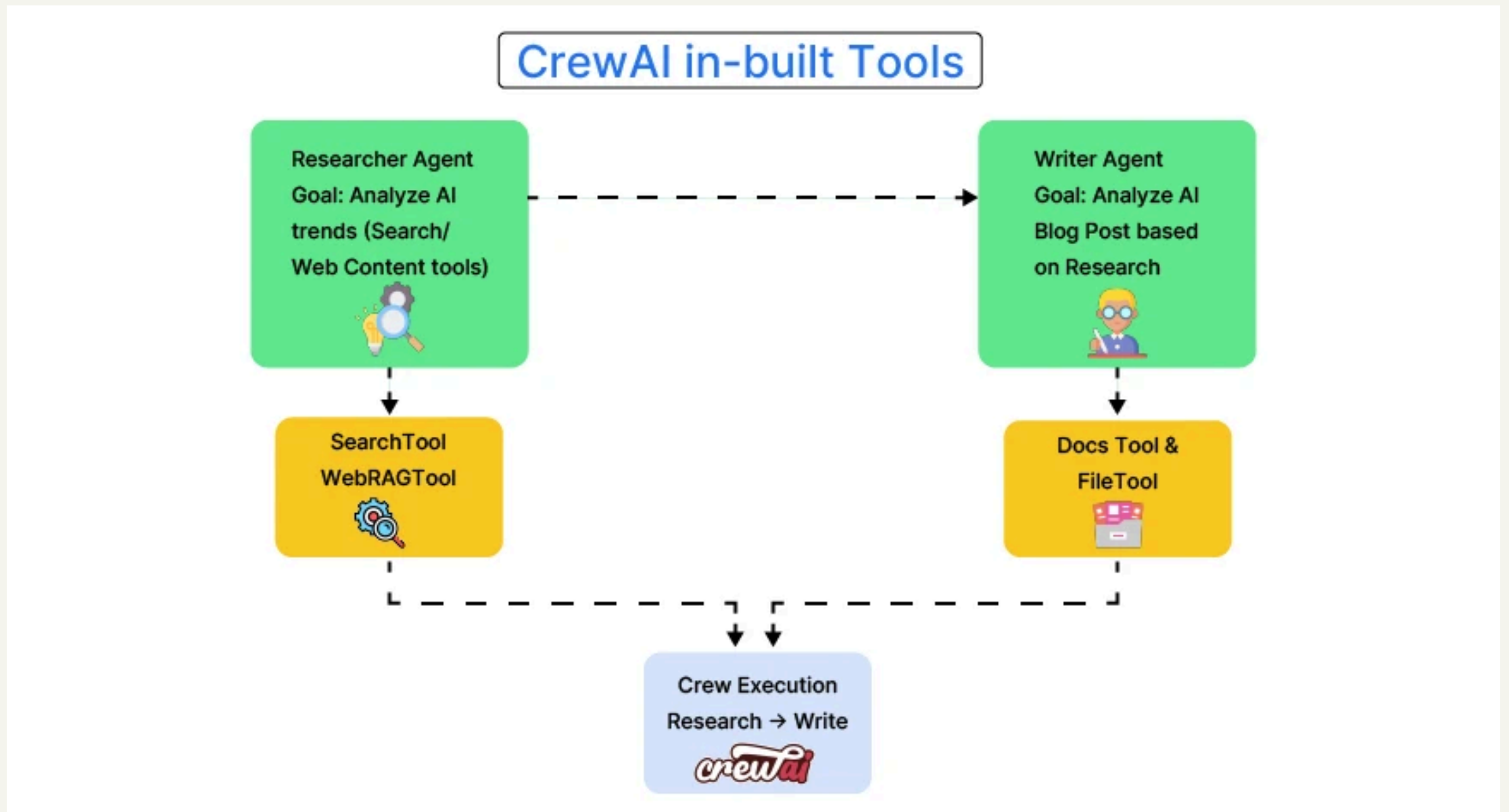
- **Modularization of Tasks:** Instead of relying on a monolithic AI model that tries to handle everything, the system breaks down user prompts and assigns them to specific tools (A, B, C)
- **Specialized Tools for Diverse Tasks:** Each tool specializes in a distinct capability, which makes the overall system more efficient and scalable. E.g Tool A could be web search, Tool B could be a code executor
- **Sequential Processing:** The model runs sequential queries through the tools, which means that multiple prompts can be processed one by one, and each tool independently depending on the task and LLM's reasoning

Tool Use Pattern – Flow



- **Step 1 – Input Task:** The user starts by asking “What is 2 times 3?”
- **Step 2 – Interpretation:** The LLM recognizes this as a mathematical operation based on its reasoning
- **Step 3 – Tool Selection:** The LLM selects the multiply tool
- **Step 4 – Payload Creation:** The LLM extracts relevant arguments (a: 2 and b: 3), prepares a payload of the tool name and tool arguments
- **Step 5 – Execution:** Usually an agent executor will leverage the tool call payload and call the relevant tool and get the output and pass it back to the LLM
- **Step 6 – Result:** The LLM analyzes the result and decided to call further tools if necessary or returns the result if it answers the input task successfully.

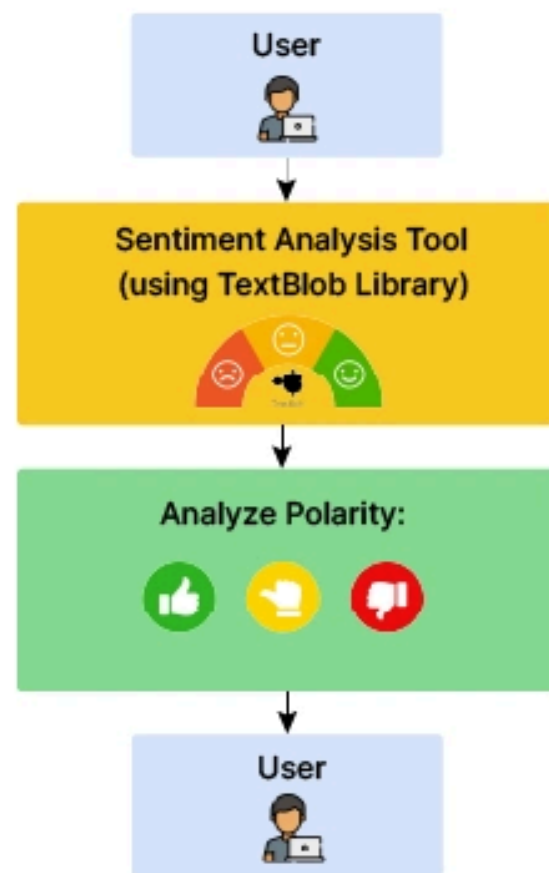
Hands-on Agentic AI Systems with Tool Use



- We build a Blog Research and Content Generation Agent (BRCGA) that automates the process of researching the latest trends in the AI industry and crafting high-quality blog posts
- The BRCGA is divided into two core roles:
 - **Researcher Agent:** Focused on gathering insights and market analysis.
 - **Writer Agent:** Responsible for creating well-written blog posts based on the research.
- Researcher Agent uses search tools to get detailed information on topics from the web
- Writer Agent uses document and file tools to read and write the article into relevant files once generated
- Check out the linked article for detailed hands-on implementation

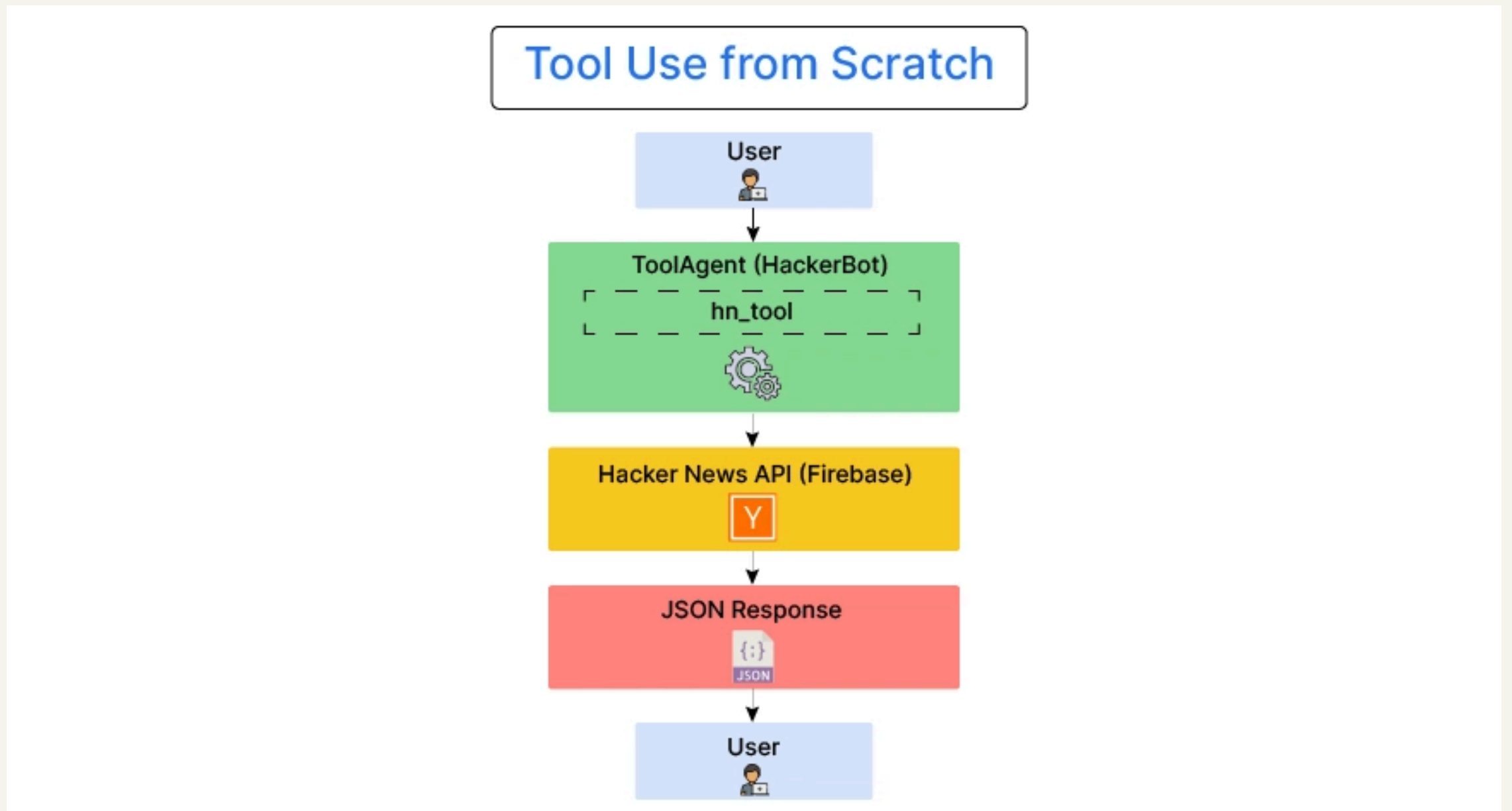
Hands-on Agentic AI Systems with Tool Use

Custom Tool Using CrewAI



- We built SentimentAI, an agent, designed to act as a powerful assistant that analyses text content, evaluates its sentiment, and ensures positive and engaging communication and generated content.
- **Agent's Purpose:**
 - **Text Analysis:** Evaluate the tone of messages, emails, social media posts, or any other form of written communication.
 - **Sentiment Monitoring:** Identify positive, negative, or neutral sentiments to help users maintain a positive engagement.
 - **User Feedback:** Provide actionable insights for improving communication by suggesting tone adjustments based on sentiment analysis.
- This was used on a custom example of generating emails for sales leads
- Check out the linked article for detailed hands-on implementation

Hands-on Agentic AI Systems with Tool Use



- **HackerBot is an AI agent designed to fetch and present the latest top stories from Hacker News, a popular news platform focused on technology, startups, and software development**
- **This agent is built from scratch thanks to detailed examples provided by Michaelis Trofficus**
- **A custom tool is built to leverage Hacker News API to fetch live news articles**
- **An agent is built from scratch using a custom built tool decorator and a ToolAgent leveraging an LLM to decide when to call the search tool to retrieve relevant news based on the users request**
- **Credits go to Michaelis for implementing these modules from scratch in his repo**

Detailed Article

[Home](#) > [Advanced](#) > [What is Agentic AI Tool Use Pattern?](#)

What is Agentic AI Tool Use Pattern?



[Pankaj Singh](#)

Last Updated : 22 Oct, 2024



18 min read

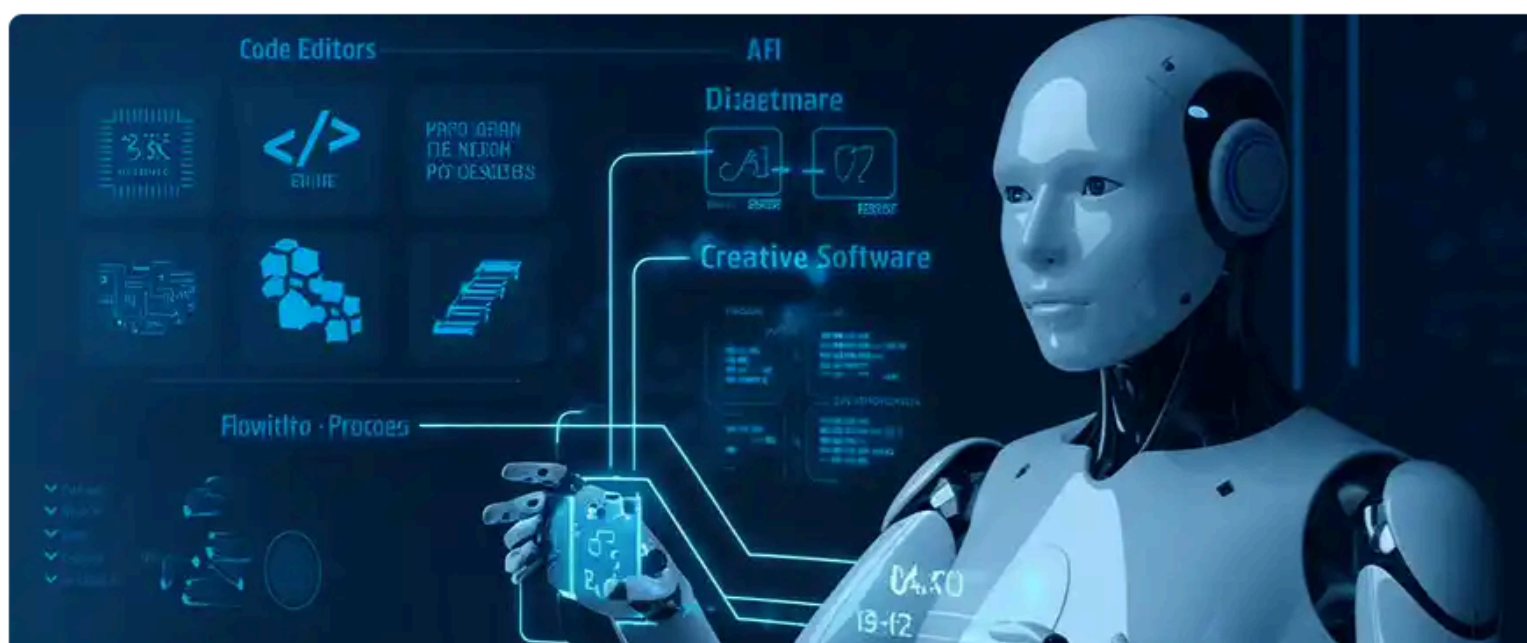


Our previous guide discussed the first Agentic AI design pattern from the Reflection, Tool Use, Planning, and Multi-Agent Patterns list. Now, we will talk about the Tool Use Pattern in Agentic AI.

Firstly, let us reiterate the Reflection Pattern article: That article sheds light on how LLMs can use an iterative generation process and self-assessment to improve output quality. The core idea here is that the AI, much like a course creator revising lesson plans, generates content, critiques it, and refines it iteratively, improving with each cycle. The Reflection Pattern is particularly useful in complex tasks like text generation, problem-solving, and code development. In the reflection pattern, the AI plays dual roles: creator and critic. This cycle repeats, often until a certain quality threshold or stopping criterion is met, such as a fixed number of iterations or an acceptable level of quality. The comprehensive article is here: [What is Agentic AI Reflection Pattern?](#)

Now, let's talk about the **Tool Use Pattern** in Agentic AI, a crucial mechanism that enables AI to interact with external systems, APIs, or resources beyond its internal capabilities.

Also, here are the 4 Agentic AI Design Pattern: [Top 4 Agentic AI Design Patterns for Architecting AI Systems](#).



CHECK OUT THE
DETAILED ARTICLE
HERE