



## **UE22AM343BB5**

### **Agentic Workflow**

---

**Dr. Shylaja S S**  
Director of Cloud Computing & Big Data (CCBD), Centre  
for Data Sciences & Applied Machine Learning (CDSAML)  
Department of Computer Science and Engineering  
[shylaja.sharath@pes.edu](mailto:shylaja.sharath@pes.edu)

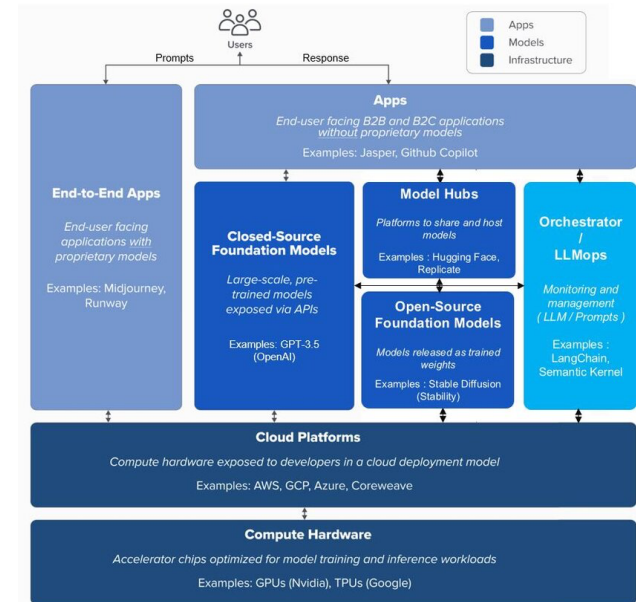
**Ack: Ujjwal MK,**  
**Teaching Assistant**

# UE22AM343BB5: Large Language Models and Their Applications

## The Foundational AI Stack: A Layered Perspective

Artificial Intelligence can be conceptualized as a stack of interconnected layers. At the base, **Semiconductors** provide the necessary computational power for AI models. Above this lies the **Cloud Infrastructure**, offering scalable resources for data storage and processing, exemplified by platforms like Snowflake. The crucial upper layer comprises **Applications**, where AI algorithms are translated into practical solutions addressing real-world problems.

While technological advancements in the lower layers are vital, the true transformative potential of AI is realized through the development and deployment of innovative applications. This layered perspective helps us understand the dependencies and opportunities within the broader AI ecosystem.



# UE22AM343BB5: Large Language Models and Their Applications

## The Impact of Generative AI: Accelerating Innovation

Generative AI represents a significant advancement in the field, enabling the rapid development and iteration of machine learning models. This acceleration allows research teams and developers to create functional prototypes in significantly shorter timeframes, moving from months to potentially just days or weeks.

This capability fosters a culture of rapid experimentation, where multiple hypotheses can be tested and refined efficiently. While the speed of prototyping has increased dramatically, it's crucial to emphasize the importance of rigorous evaluation and responsible development practices to ensure the reliability and ethical implications of these rapidly created AI solutions are thoroughly considered

### Generative AI Use Cases

Language	Visual	Auditory
Marketing (content)	Video Generation	Music Generation
Note Taking	3D Models	Voice Generation
Gene Sequencing	Design	
Code Development	Image Generation	
Essay Generation		

Traditional interactions with AI often involve a single prompt leading to a direct response. **Agentic Workflows** represent a more sophisticated approach where AI systems, known as agents, engage in iterative and deliberative processes to accomplish complex tasks. This involves multiple steps such as research, planning, execution, and refinement, mirroring human problem-solving methodologies.

This shift towards agentic workflows often yields significantly improved outcomes, particularly for tasks requiring nuanced reasoning and multiple stages of processing. The concept of an "agentic orchestration layer" is also emerging to facilitate the development and management of these complex AI agent-based applications.

# UE22AM343BB5: Large Language Models and Their Applications

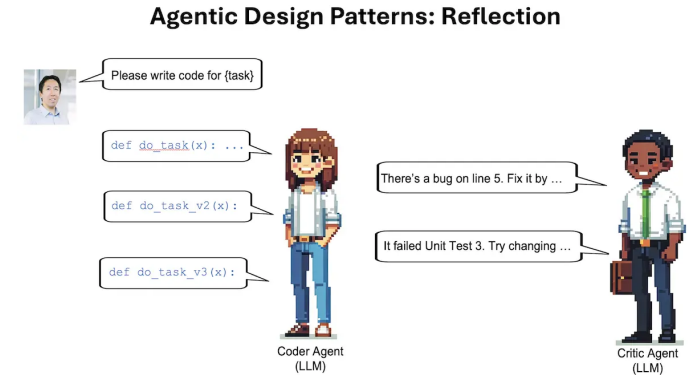
## Introducing Agentic Workflows: A Paradigm Shift in AI Interaction

	Level 1: Output Decisions Ability to make <b>decisions</b> based on natural language	Level 2: Task Decisions Can <b>choose</b> which tasks and tools to execute	Level 3: Process Decisions Can <b>create</b> new tasks and tools to execute
AI Workflow	✓	✗	✗
Router Workflow	✓	✓	✗
Autonomous Agent	✓	✓	✓

vellum

The **Reflection** design pattern equips AI agents with the ability to evaluate and critique their own outputs. Similar to peer review in academic settings or debugging in software development, this process allows the AI to identify errors, inefficiencies, or areas for improvement.

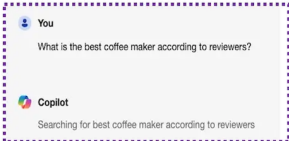
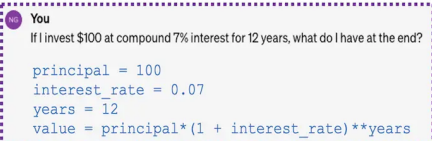
Through iterative cycles of self-assessment and revision, AI agents employing reflection can achieve more accurate and higher-quality results. This can involve a single agent reflecting on its work or a multi-agent system where one agent generates content and another provides constructive criticism.



The **Tool Use** design pattern enables AI agents to extend their functionality by leveraging external tools and APIs. These tools can range from information retrieval systems like web search engines to computational tools like code interpreters, and even interfaces for interacting with real-world applications.

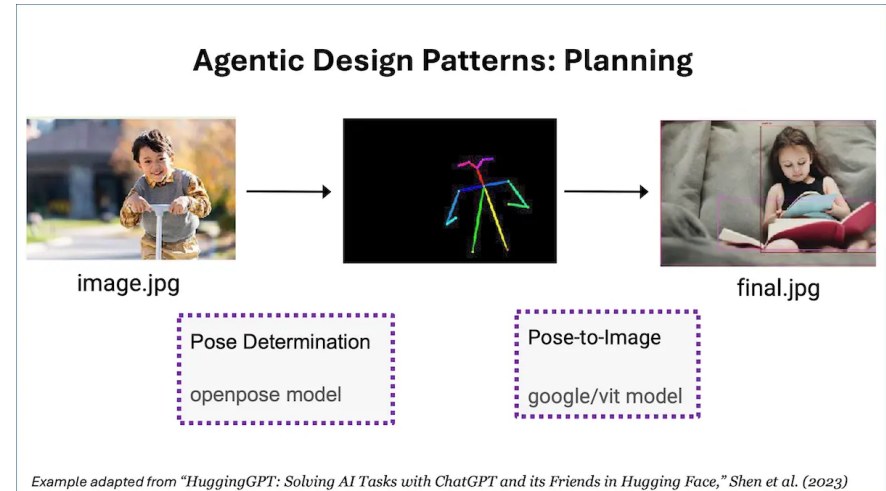
By integrating with these external resources, AI agents become more versatile and capable of addressing a wider range of complex tasks that require interaction with the external environment or specialized functionalities.

### Agentic Design Patterns: Tool Use

Web search tool	Code execution tool
 <p>Example from Bing CoPilot</p>	 <p>Example from ChatGPT</p>

The **Planning** design pattern involves AI agents breaking down complex requests or goals into a sequence of smaller, more manageable sub-tasks. This decomposition allows the agent to approach intricate problems in a structured and organized manner, similar to project management methodologies.

By planning the necessary steps to achieve a complex objective, AI agents can better manage the inherent complexity and increase the likelihood of successful task completion. This planning process can be static, where the entire plan is defined upfront, or dynamic, where the agent adjusts its plan based on new information or unexpected events.

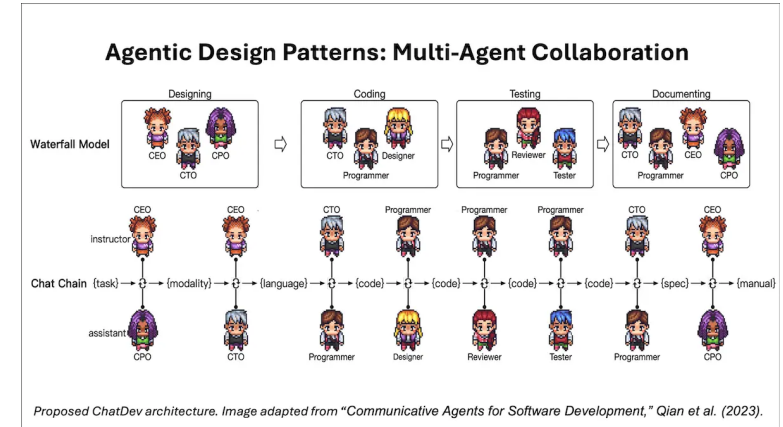


Credits: <https://www.deeplearning.ai/the-batch/agentic-design-patterns-part-4-planning/>



**Multi-Agent Collaboration** is a design pattern that leverages the combined capabilities of multiple AI agents working together to achieve a common goal. In this model, different agents are assigned specialized roles or functionalities, allowing them to contribute their unique expertise to the task at hand.

These agents can communicate and interact with each other, exchanging information and coordinating their efforts to solve complex problems, much like a team of researchers or engineers working on a project. This collaborative approach can lead to enhanced performance and the ability to tackle more intricate challenges.



Beyond agentic workflows, advancements in **Visual AI** are significantly expanding the scope of AI applications. Visual AI encompasses technologies that enable computers to interpret and understand visual information from images and videos.

Applications of visual AI are diverse and span numerous fields, including autonomous vehicles, medical imaging analysis, quality control in manufacturing, and retail analytics. The ability to effectively process and understand visual data, along with other forms of unstructured data like audio and text, is crucial for building more powerful and versatile AI systems.

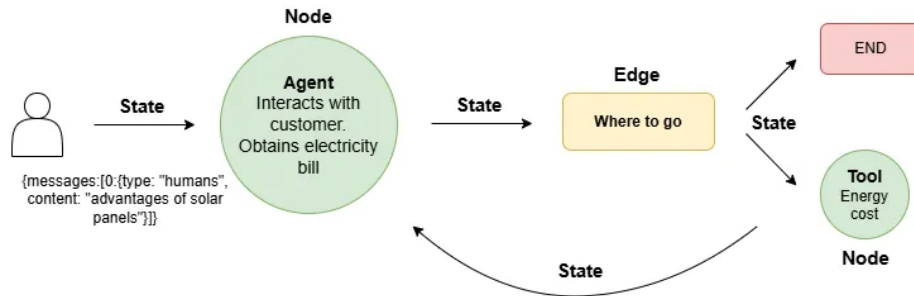


The development of Agentic AI builds upon the concept of compound LLMs. This approach involves strategically linking multiple LLMs in sequence to enhance task outcomes. For example, a system might utilize one LLM for initial drafting, a second for critical review, and a third for revision based on the critique, reflecting an iterative refinement process.

Compound LLM systems demonstrate the value of combining specialized AI functions. By structuring interactions between LLMs, such systems can achieve a level of quality and sophistication in their output that often surpasses what a single LLM can produce alone. This represents a step towards AI systems capable of reflection and improvement during task execution.

Agentic AI expands significantly on compound LLMs by employing a diverse set of "agents." These agents are not limited to LLMs; they can be any computational tool or resource, such as data retrieval systems or external APIs, chosen for specific capabilities. A key element is the presence of an 'orchestrator' agent.

The orchestrator dynamically manages the workflow, selecting and directing the appropriate agents or tools based on the current state of the task and its requirements. This contrasts with rigid, pre-defined workflows, allowing the system to adapt its approach in response to intermediate results or changing conditions, leading to more flexible and robust problem-solving.



Credits: <https://medium.com/pythoneers/building-ai-agent-systems-with-langgraph-9d85537a6326>

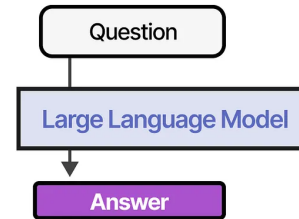
# UE22AM343BB5: Large Language Models and Their Applications

## Foundational Concept - Compound LLMs

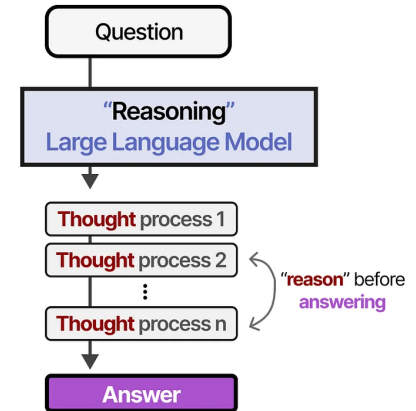
The development of Agentic AI builds upon the concept of compound LLMs. This approach involves strategically linking multiple LLMs in sequence to enhance task outcomes. For example, a system might utilize one LLM for initial drafting, a second for critical review, and a third for revision based on the critique, reflecting an iterative refinement process.

Compound LLM systems demonstrate the value of combining specialized AI functions. By structuring interactions between LLMs, such systems can achieve a level of quality and sophistication in their output that often surpasses what a single LLM can produce alone. This represents a step towards AI systems capable of reflection and improvement during task execution.

**"Regular" LLMs**



**"Reasoning" LLMs**



The core advantage of Agentic AI lies in its ability to effectively coordinate multiple specialized agents. This allows for the decomposition of complex problems, assigning sub-tasks to the agents best suited for them. Such orchestration enhances overall system intelligence, efficiency, and the potential for generating novel solutions by combining diverse functionalities.

Furthermore, the inherent adaptability of Agentic AI systems is a key differentiator. These systems can modify their strategies based on real-time feedback or changing environmental factors, making them more resilient and effective in dynamic, real-world situations compared to less flexible AI architectures.

### Conclusion: The Future of Intelligent Systems

---

The field of Artificial Intelligence is undergoing rapid transformation, with a significant emphasis on developing practical and impactful applications. The emergence of agentic workflows, driven by key design patterns, and the advancements in visual AI are paving the way for more sophisticated and capable intelligent systems.

As AI continues to evolve, these trends will unlock new possibilities for innovation across a wide range of industries and research domains. Understanding these fundamental concepts is crucial for future researchers, developers, and anyone interested in the transformative potential of artificial intelligence.



## **UE22AM343BB5**

### **Agentic Workflow**

---

**Dr. Shylaja S S**  
Director of Cloud Computing & Big Data (CCBD), Centre  
for Data Sciences & Applied Machine Learning (CDSAML)  
Department of Computer Science and Engineering  
[shylaja.sharath@pes.edu](mailto:shylaja.sharath@pes.edu)

**Ack: Ujjwal MK,**  
**Teaching Assistant**





# **LARGE LANGUAGE MODEL**

**UE21CS421AC1**

**AutoGen, CrewAI**

---

**Dr. Shylaja S S**

Department of Computer Science and Engineering

# LARGE LANGUAGE MODEL

---

## AutoGen, CrewAI – Features

Department of Computer Science and Engineering

### 1. *Multi-Agent Communication Framework*

- AutoGen allows you to define multiple AI agents (and humans!) that can talk to each other via structured messages.
- Each agent has a role and behavioral logic, like "Coder", "Reviewer", or "Planner".

### 2. *Conversational Workflow Orchestration*

- It supports looping dialogues between agents until a task is complete.
- This enables powerful coordination for tasks like code generation, debugging, or multi-step problem solving.

### 3. *Extensibility & Tool Integration*

- Agents can be connected to external tools, functions, APIs, or custom code.
- You can easily create agents that call Python code, run shell commands, or interact with web APIs.

```
from autogen import UserProxyAgent, AssistantAgent

# Create a user-facing agent
user_proxy = UserProxyAgent(name="user_proxy", human_input_mode="TERMINAL")

# Create a code assistant agent
code_assistant = AssistantAgent(name="code_assistant", llm_config={"config_list": [...]})

# Let them chat
user_proxy.initiate_chat(
    code_assistant,
    message="Can you write a Python function to calculate factorial?"
)
```

```
from autogen import GroupChat, GroupChatManager

# Add another agent for reviewing
reviewer = AssistantAgent(name="reviewer", llm_config={"config_list": [...]}

# Group chat setup
groupchat = GroupChat(agents=[user_proxy, code_assistant, reviewer], messages=[], max_round=5)
manager = GroupChatManager(groupchat=groupchat, llm_config={"config_list": [...]})

# Start a multi-agent loop
user_proxy.initiate_chat(manager, message="Please build and review a function to sort a list of numbers.")
```

# LARGE LANGUAGE MODEL

## AutoGen – Example

---



```
from autogen import PythonCodeExecutorAgent

# An agent that can execute Python code
executor = PythonCodeExecutorAgent(name="executor", human_input_mode="NEVER")

# Let user_proxy chat with executor
user_proxy.initiate_chat(
    executor,
    message="Write and execute a script that prints the Fibonacci sequence up to 100."
)
```

### 1. *Modular Agent Roles with "Crew" Structure*

- You define a crew composed of specialized agents, like an Analyst, Researcher, or Presenter.
- Each agent has a clear task description, improving task division and collaboration.

### 2. *Task Planning & Delegation*

- CrewAI supports auto-task planning, where a Manager agent can split tasks and assign them to other agents.
- This enables hierarchical workflows, like planning → researching → writing.

### 3. *Deterministic & Reusable Pipelines*

- You can run agents in a defined sequence, like a production line.
- Perfect for use cases like content creation, document processing, or automated research workflows.

# LARGE LANGUAGE MODEL

## CrewAI – Example

---



```
from crewai import Agent, Task, Crew
from langchain.chat_models import ChatOpenAI # Or use any other supported LLM

# LLM Setup (you can replace this with HuggingFace or Ollama if needed)
llm = ChatOpenAI(temperature=0.7, model="gpt-4") # Or "gpt-3.5-turbo"

# Define Agents
researcher = Agent(
    role="Research Specialist",
    goal="Find up-to-date information on quantum computing",
    backstory="You are an expert researcher who always finds accurate and current info.",
    verbose=True,
    llm=llm,
)
```



# LARGE LANGUAGE MODEL

## CrewAI – Example

---



```
summarizer = Agent(  
    role="Scientific Writer",  
    goal="Summarize complex research into simple, readable content",  
    backstory="You write blog posts that help beginners understand tough topics.",  
    verbose=True,  
    llm=llm,  
)
```

# Define Tasks

```
task1 = Task(  
    description="Search and gather recent breakthroughs in quantum computing.",  
    agent=researcher,  
)
```

```
task2 = Task(  
    description="Summarize the findings from the researcher in a blog-post format.",  
    agent=summarizer,  
    depends_on=[task1],
```

# LARGE LANGUAGE MODEL

## CrewAI – Example

---



```
# Define the Crew (the pipeline)
crew = Crew(
    agents=[researcher, summarizer],
    tasks=[task1, task2],
    verbose=True,
)
```

```
# Kick off the workflow
result = crew.kickoff()
print(" Final Output:\n", result)
```



**THANK YOU**

---

**Dr. Shylaja S S**

Department of Computer Science Engineering

**[shylaja.sharath@pes.edu](mailto:shylaja.sharath@pes.edu)**