Pre-Read Document

Agentic AI and Agentic RAG

This pre-read introduces the foundations of AI agents, their evolution, core architectural components, and how they integrate with retrieval systems. It prepares you for discussions on agent design, orchestration, and multi-agent workflows.

---

1. Evolution of AI Agents

AI systems have evolved through several stages:

1. **Rule-Based Systems**

   Predefined logic and deterministic behavior. No learning capability.

2. **Machine Learning Models**

   Predictive systems trained on data but limited to single-task outputs.

3. **LLM-Based Systems**

   Capable of reasoning, language understanding, and contextual generation.

4. **Agentic Systems**

   Systems that can:

   - Plan multi-step actions
   - Use tools
   - Retrieve knowledge

    - Maintain memory
    - Self-correct

Agentic AI represents the shift from “generate a response” to “decide and act toward a goal.”

---

2. Core Components of an AI Agent

An AI agent typically includes:

2.1 LLM (Reasoning Engine)

Handles:

- Interpretation
- Planning
- Decision-making

2.2 Tools

External capabilities such as:

- APIs
- Databases
- Search systems
- Code execution

2.3 Memory

Stores:

- Conversation history
- Task state
- Persistent knowledge

2.4 Planner

Breaks complex tasks into steps.

2.5 Executor

Executes actions and collects outputs.

An agent operates in a loop:

Understand → Plan → Act → Observe → Repeat

---

3. Planning, Reasoning, and Tool Usage

Planning

Agents break down high-level goals into smaller tasks.

Approaches:

- Step-by-step reasoning
- Task decomposition

- Hierarchical planning

Reasoning

Agents use reasoning patterns such as:

- Chain-of-thought reasoning
- Reflection
- Self-correction
- Multi-step analysis

Tool Usage

Agents extend their capabilities by calling tools.

Example workflow:

1. Identify missing information
2. Select relevant tool
3. Execute tool
4. Integrate result into reasoning

Tool integration is essential for grounded and reliable systems.

---

4. OpenAI Agents SDK (Conceptual Overview)

The OpenAI Agents SDK enables developers to build structured, tool-aware AI agents.

Key ideas:

- Define tools
- Define agent behavior
- Manage memory
- Handle execution flow

The SDK abstracts:

- Tool calling
- Context management
- Orchestration logic

It simplifies building production-ready agent systems.

---

5. Agent Architectures

5.1 ReAct Architecture

Reason + Act loop:

- Generate reasoning
- Call tool
- Observe result
- Continue

5.2 Planner–Executor Architecture

Separate components:

- Planner creates plan
- Executor performs actions

Improves modularity and reliability.

5.3 Reflection-Based Architecture

Agent:

- Produces answer
- Evaluates output
- Refines response

Reduces hallucinations and logical errors.

5.4 Retrieval-Augmented Agents

Combines:

- RAG pipeline
- Planning loop

Agent retrieves knowledge dynamically before responding.

---

6. CRAG (Corrective RAG)

Corrective RAG improves standard RAG by adding a verification layer.

Key idea:
Evaluate retrieved documents before generation.

CRAG introduces:

- Retrieval evaluation
- Filtering low-quality chunks
- Correction mechanisms
- Iterative retrieval if necessary

This reduces hallucinations and improves answer grounding.

---

7. Agentic AI and Agentic RAG

Agentic AI

An AI system that:

- Pursues goals
- Makes decisions
- Uses tools
- Maintains memory
- Adapts over time

Agentic RAG

A RAG system enhanced with:

- Planning
- Multi-step retrieval
- Iterative refinement
- Self-correction

Instead of single retrieval + generation, the system can:

1. Retrieve
2. Evaluate
3. Retrieve again if needed
4. Generate final answer

---

8. Single-Agent vs Multi-Agent Systems

Single-Agent Systems

One agent:

- Handles planning
- Uses tools
- Generates final output

Advantages:

- Simpler design
- Easier debugging

Limitations:

- Limited specialization

- Harder to scale complex tasks

Multi-Agent Systems

Multiple agents collaborate:

Examples:

- Research agent
- Critic agent
- Execution agent
- Validation agent

Advantages:

- Specialization
- Parallelism
- Modular reasoning

Challenges:

- Coordination complexity
- Communication overhead

---

9. Building a Simple AI Agent (Conceptual Flow)

Basic steps:

1. Define goal
2. Define tools
3. Implement reasoning loop
4. Capture observations
5. Iterate until completion

Core pattern:

Input → Plan → Tool Call → Observe → Decide → Respond

---

10. Multi-Agent Workflows with LangGraph

LangGraph enables:

- State-based workflows
- Node-based execution
- Directed graphs of agents

Conceptually:

- Each node = agent or function
- Edges define execution flow
- State shared across nodes

Useful for:

- Structured workflows
- Complex orchestration
- Multi-step pipelines

---

11. Different Types of Agents

11.1 Reactive Agents

Respond directly to input without planning.

11.2 Deliberative Agents

Plan before acting.

11.3 Tool-Using Agents

Call APIs and external systems.

11.4 Retrieval Agents

Integrate RAG capabilities.

11.5 Reflective Agents

Self-evaluate and improve responses.

---

12. Agentic Handoff

Agentic handoff occurs when:

- One agent transfers responsibility to another
- Task context is passed along
- Specialized processing continues

Example:

- Research agent gathers information
- Analysis agent processes findings
- Report agent formats output

Proper handoff requires:

- Shared state
- Clear task boundaries
- Structured communication

Preparation Before Session

You should be comfortable with:

- Basic RAG pipeline
- Tool calling concepts
- LLM reasoning patterns
- State and workflow modeling

This preparation will enable meaningful discussion on designing, orchestrating, and scaling agentic systems in real-world environments.