Introduction to Large Language Models

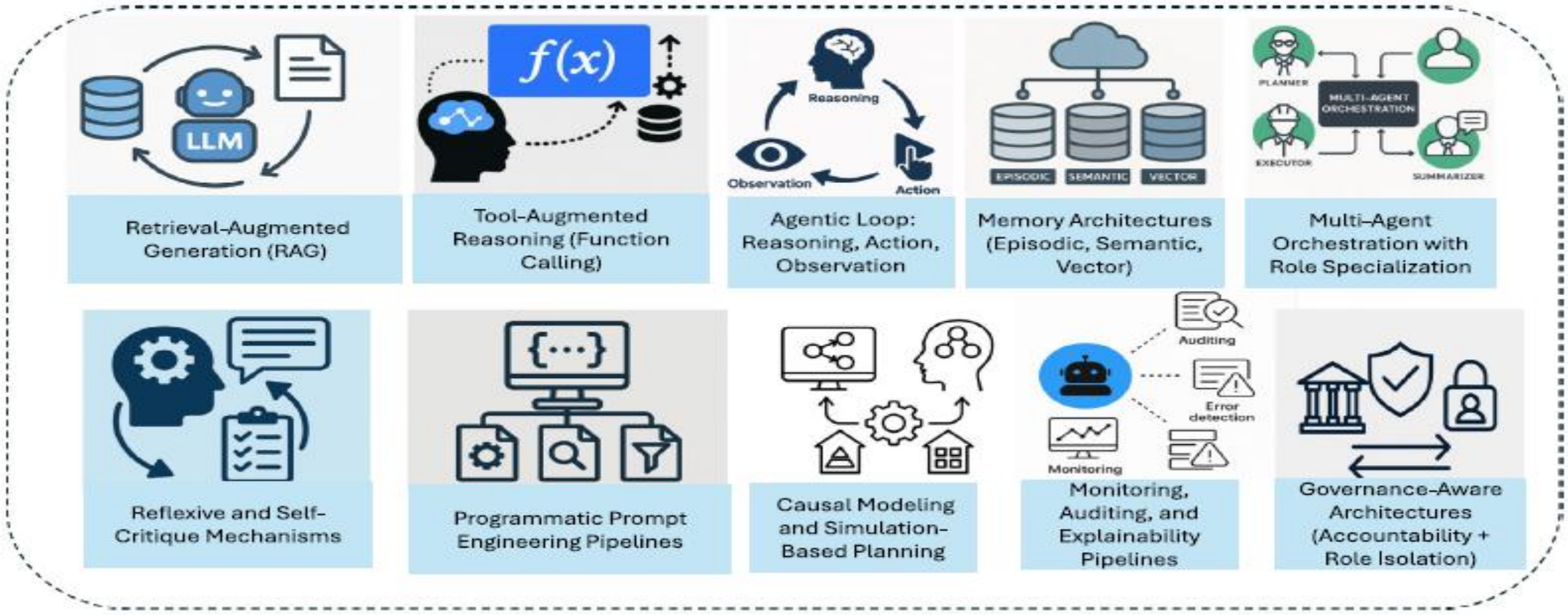
Spring 2026

Agentic AI II

Planning, Memory, Agent Architecture Types, Agentic AI Use Cases

(Some slides adapted from Ralph Grishman at NYU,
Yejin Choi at UWashington, N. Tomura at UDepaul, Jurafsky and Martin,
CS224N, CS224, CME295 at Stanford and other resourses on the web)

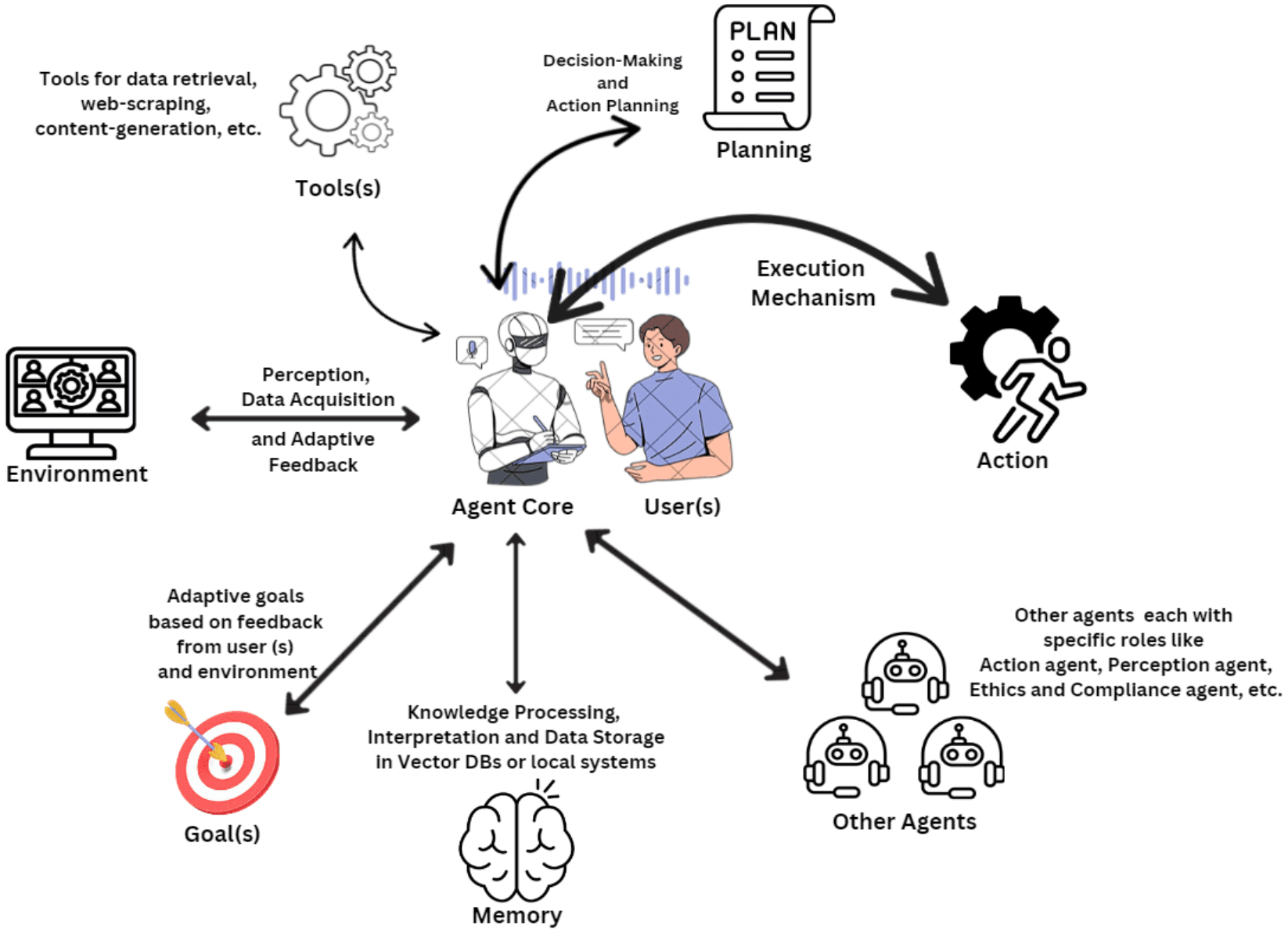
Evolving architectural and algorithmic mechanisms



Ten evolving architectural and algorithmic mechanisms such as RAG, tool augmentation, dynamic memory, causal modeling, orchestration, and reflexive self-evaluation are shown as key enablers to advance beyond prior usage toward addressing current limitations in reliability, scalability, and explainability. These techniques, while previously applied in isolated agent systems, are here recontextualized to support the demands of modern AI Agents and Agentic AI, enabling coordinated, adaptive, and verifiable behavior in increasingly complex and dynamic environments.

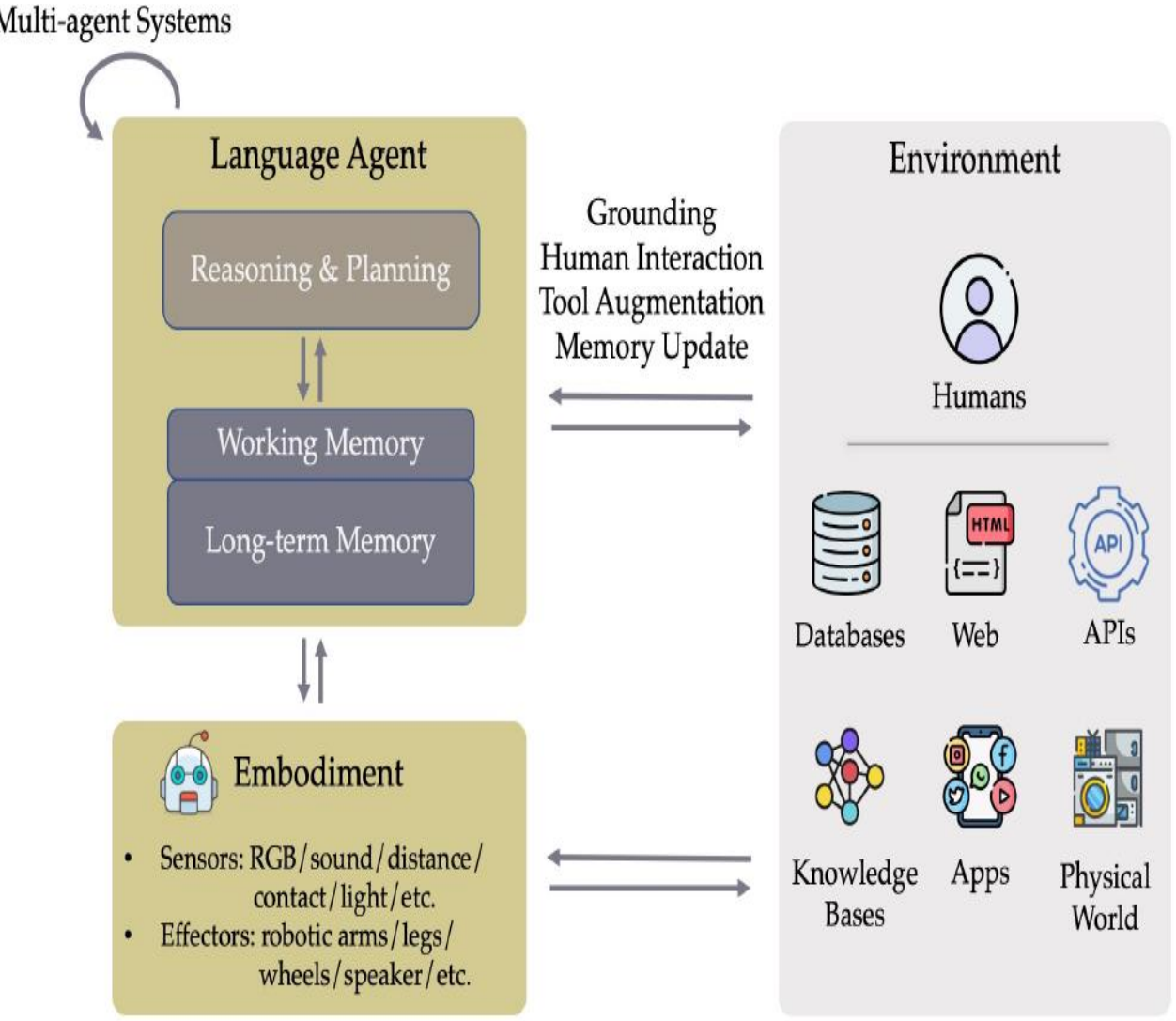
Agentic AI

- Agentic AI is action-oriented, designed to make decisions, plan tasks, and execute them autonomously in dynamic environments.
- It does not just generate responses but actively interacts with its surroundings, making strategic choices based on goals and real-time feedback.



https://learnopencv.com/wp-content/uploads/2025/02/Agentic-AI-Workflow-1.png

Language Agents



Language Agents: Foundations, Prospects, and Risks, Yu Su et al.

Figure 1: A conceptual framework for language agents.

The **Language Agent** is the central decision-making unit. It is typically powered by a **Large Language Model (LLM)**. It combines LLM-based reasoning, memory, tools, and embodiment to perceive, plan, and act in real environments—enabling agentic and multi-agent AI systems.

a. Reasoning & Planning : This is the "thinking" unit

b. Working Memory : Stores **short-term context**

c. Long-Term Memory: Stores Persistent knowledge across interactions, Past experiences, user preferences, learned facts

Grounding and Interaction with the Environment

a. Grounding: Connecting abstract language to real-world meaning

b. Human Interaction : Humans can give instructions, feedback, or corrections. This enables **human-in-the-loop** agentic systems.

c. Tool Augmentation : The agent can use tools such as: APIs, Search engines, Databases, Software applications

d. Memory Update : Results of actions and interactions flow back into: memory.

Embodiment (Optional but Powerful)

The **Embodiment** layer represents agents that have a physical or simulated body.

a. Sensors (Perception) : RGB (vision), Sound (audio), etc. Sensors provide **raw observations** from the environment.

b. Effectors (Action) : Robotic arms, Legs or wheels, Speakers

Effectors allow the agent to **physically act**, not just digitally respond.

Environment (Where Actions Happen) : Humans, Digital Resources, . Physical World (Objects, real-world Dynamics)

Multi-Agent Systems (Why the Title Matters)

Although a single agent is drawn, the title emphasizes **Multi-agent Systems**: Examples: A planner agent coordinating with executor agents, Research, coding, and evaluation agents working together

Planning in Agentic AI

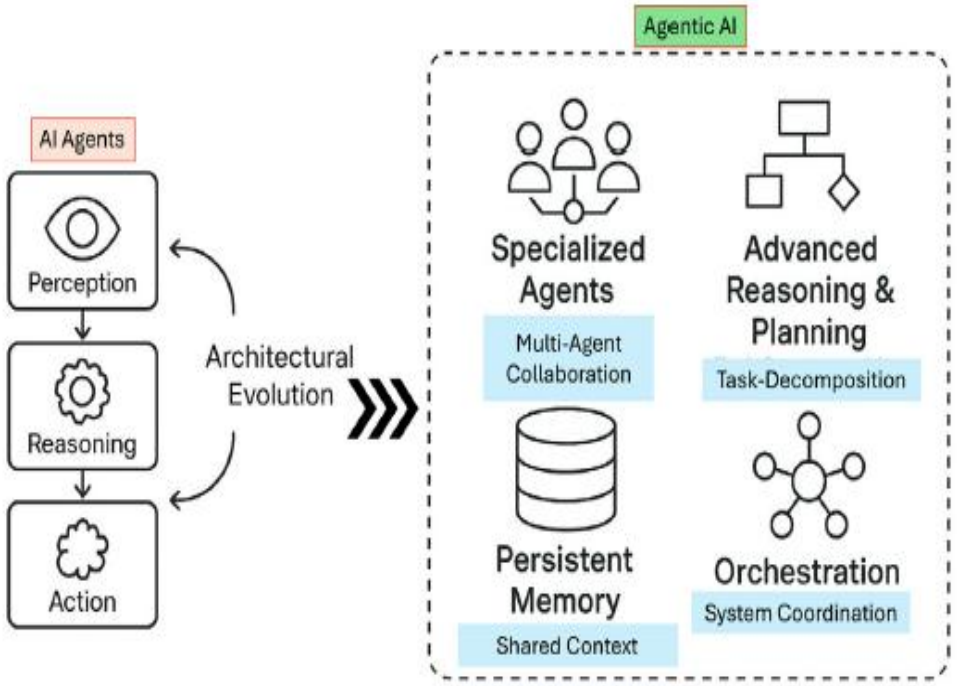
Planning is a core capability in *agentic AI systems*—systems that can autonomously pursue goals, make decisions, and take actions over time. In agentic AI, planning answers the question:

Given a goal and the current state of the world, what sequence of actions should the agent take to achieve that goal?

In agentic AI, planning refers to the process of:

1. **Goal specification** – defining what the agent is trying to achieve
2. **State modeling** – representing the current situation (internal + external)
3. **Action modeling** – defining what actions are possible and their effects
4. **Plan generation** – producing an ordered or conditional sequence of actions
5. **Plan execution and monitoring** – carrying out actions and observing outcomes
6. **Replanning** – adapting when the environment or assumptions change

Unlike reactive systems, *agentic* systems explicitly reason about the future.



Types of Planning in Agentic AI

1. Task Decomposition (Hierarchical Planning)

Agents break high-level goals into subgoals:

```
Goal: Write a research paper
  ├─ Review literature
  ├─ Design methodology
  ├─ Run experiments
  ├─ Write draft
  └─ Revise and submit
```

Often called:

- **Hierarchical Task Networks (HTN)**
- **Plan-and-execute** or **Think–Act loops**

LLMs are very effective at this form of planning

2. Deliberative Planning

The agent reasons explicitly before acting:

- Generates a full or partial plan
- Evaluates alternatives
- Executes actions sequentially

Typical loop:

```
Observe → Plan → Act → Observe → Replan
```

Used when:

- Actions are costly
- Errors are hard to undo
- Accuracy matters more than speed

Types of Planning in Agentic AI

3. Reactive + Planning Hybrid

Agent combines:

- **Reactive policies** (fast responses)
- **Planning modules** (long-term goals)

Example:

- Autonomous robot avoids obstacles reactively
- But plans its route strategically

4. Planning with Uncertainty

Agent reasons under uncertainty using:

- Probabilistic outcomes
- Expected rewards
- Belief states

Related approaches:

- **MDPs / POMDPs**
- **Model-based reinforcement learning**

In LLM-based agents, uncertainty is often handled via:

- Replanning
- Self-correction
- Redundancy (multiple plans, voting)

Planning Architectures in Agentic Systems

1 ReAct (Reason + Act)

Agents interleave reasoning and action:

```
Thought → Action → Observation → Thought → Action
```

Planning is *incremental*, not fixed upfront.

2 Plan-and-Execute

1. Generate a full plan
2. Execute steps one by one
3. Replan if execution fails

Used in enterprise agents and workflow automation.

3 Tool-Integrated Planning

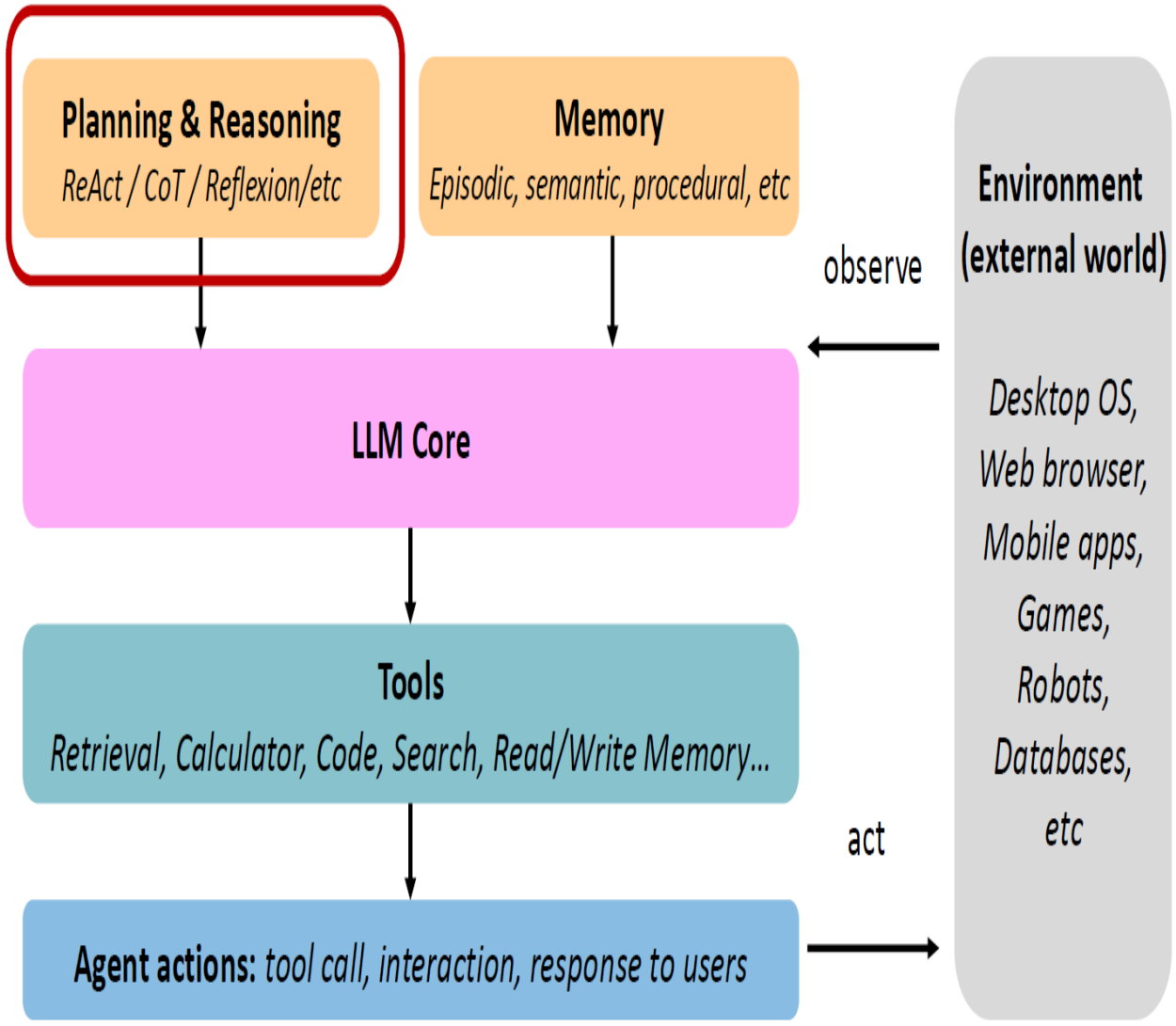
Agent plans across:

- Internal reasoning steps
- External tools (search, code, APIs)
- Human interactions

Example:

```
Plan:
1. Search papers on topic
2. Summarize key methods
3. Compare results
4. Write section draft
```

Role of LLMs in Planning



Large Language Models (LLMs) play a **central enabling role** in planning for agentic AI by providing flexible, language-based reasoning and abstraction capabilities. Rather than replacing classical planners, LLMs **augment and orchestrate planning** in complex, open-ended environments.

Key roles of LLMs in planning:

1. **Goal Interpretation**
   - Translate vague or high-level human goals into actionable objectives.
   - Clarify constraints, priorities, and success criteria.
2. **Task Decomposition**
   - Break complex goals into ordered or hierarchical sub-tasks.
   - Enable multi-step and long-horizon plans.
3. **Plan Generation**
   - Propose sequences of actions using world knowledge and common sense.
   - Create alternative plans when multiple strategies exist.
4. **Contextual Reasoning**
   - Incorporate memory, retrieved knowledge, and tool outputs into plans.
   - Adapt plans to changing environments and partial information.
5. **Plan Evaluation and Self-Critique**
   - Reflect on plan quality, feasibility, and risks.
   - Revise or improve plans before or during execution.
6. **Replanning and Adaptation**
   - Adjust plans dynamically when actions fail or new information appears.
   - Support continuous plan–act–observe loops.

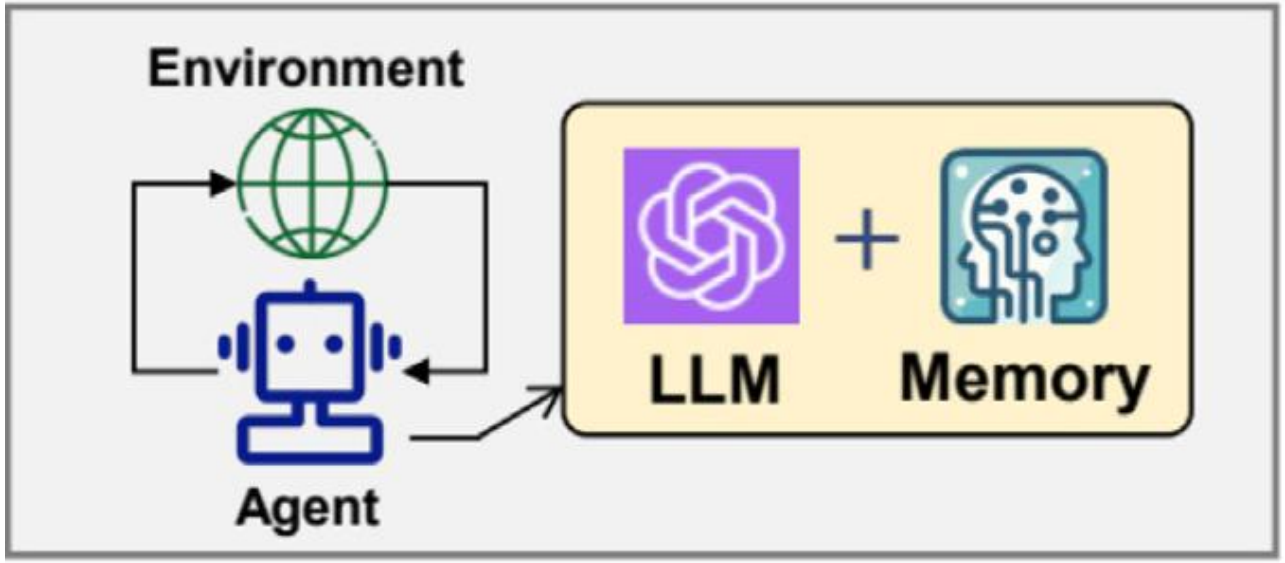
Planning in agentic AI

Summary: Planning in agentic AI is not just about choosing actions—it is about continuously reasoning, adapting, and coordinating actions toward goals in dynamic, uncertain environments.

Memory in Agentic AI

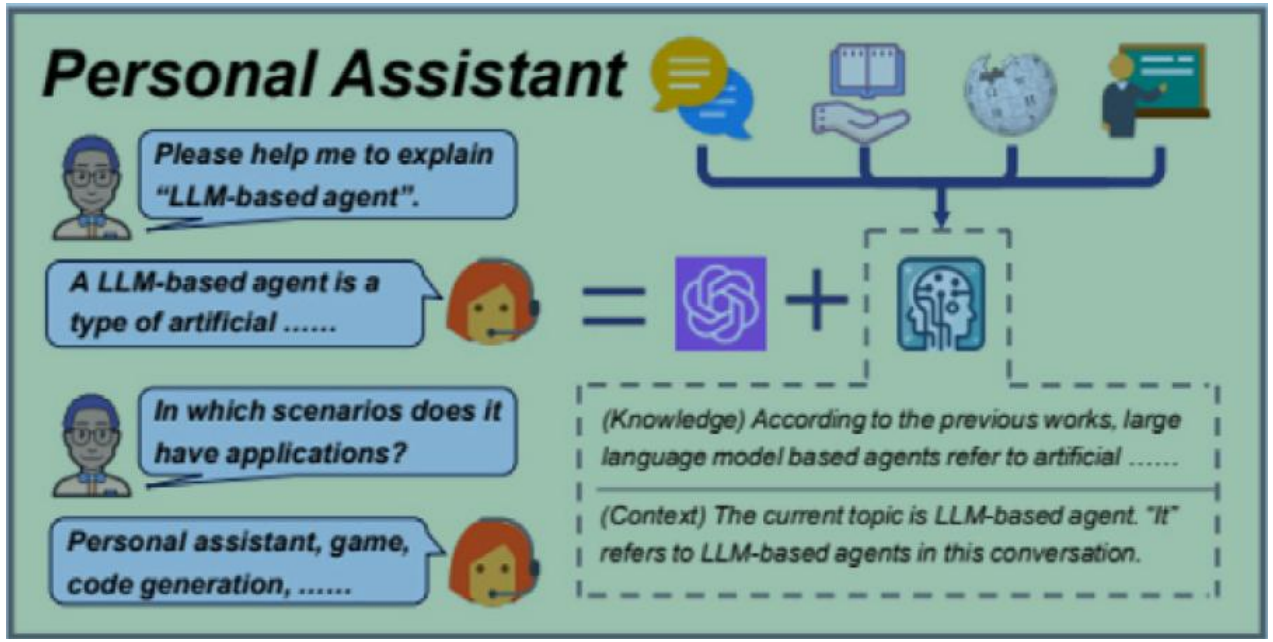
- **Memory** in agentic AI refers to an agent's ability to **store, retrieve, and use past information** to influence future reasoning, planning, and actions.
- Without memory, an agent is reactive. With memory, an agent becomes **context-aware, adaptive, and goal-persistent**.

In short: **Planning decides what to do next; memory determines what the agent knows from before.**



Agentic AI systems operate over **multiple steps, tasks, and time horizons**. Memory enables:

- Long-term goal pursuit
- Learning from past successes and failures
- Personalization and consistency
- Coordination across plans and tools
- Reduced repetition and hallucination



Core Types of Memory in Agentic AI

1 - Short-Term Memory (Working Memory)

Purpose:
Holds information relevant to the *current reasoning step or episode*.

Examples:

- Current user query
- Intermediate reasoning steps
- Recently observed tool outputs

Characteristics:

- Limited capacity
- Often implemented via the LLM context window
- Cleared or overwritten frequently

Analogous to human **working memory**

2 - Long-Term Memory

Purpose:
Stores information across time and interactions.

Examples:

- User preferences
- Past task outcomes
- Learned facts
- Previous plans

Common implementations:

- Vector databases (semantic memory)
- Structured databases (key–value memory)
- Knowledge graphs

This enables *continuity* across sessions.

Core Types of Memory in Agentic AI

3 - Episodic Memory

Purpose:
Remembers **past experiences as sequences of events**.

Examples:

- "Last time the agent tried this plan, step 3 failed"
- Past conversations with timestamps
- Logs of actions and outcomes

Use cases:

- Reflection
- Learning from mistakes
- Case-based reasoning

4 - Semantic Memory

Purpose:
Stores **general knowledge abstracted from experience**.

Examples:

- Domain facts
- Definitions
- Rules and heuristics
- Patterns learned over time

Episodic → generalized → semantic

Core Types of Memory in Agentic AI

5 - Procedural Memory

Purpose:
Remembers **how to do things**.

Examples:

- Known workflows
- Successful plans
- Tool usage patterns
- Prompt templates

This supports:

- Skill reuse
- Faster planning
- Automation

Memory Architectures in Agentic Systems

- Retrieval-Augmented Memory (RAM) : Memories are embedded, Relevant ones are retrieved via similarity search

- Layered Memory Architecture : This mirrors **cognitive architectures :** Working, Episodic, Semantic, Procedural

- Reflective Memory

Advanced agents:

- Periodically summarize experiences
- Extract lessons
- Update long-term memory

Example:

“This approach works better when constraint X is explicit.”

Remember: Context Window

- The **context window** is the maximum amount of text (measured in **tokens**) that a Large Language Model can **read, reason over, and generate from at one time**.
- The **context window size** of a Large Language Model (LLM) is the **maximum number of tokens** (words + subwords + punctuation) the model can consider **at one time** when generating a response. It limits how much information the model can "see" simultaneously.
- Larger context windows allow longer documents, conversations, and reasoning chains—but with higher compute cost.

What "Token" Means (Briefly)

- A token is not exactly a word.
- Rough heuristic:
  - **1 token ≈ 0.75 English words**
  - **1,000 tokens ≈ 700–800 words**

Typical Context Window Sizes

Category	Approximate Size
Early LLMs	2k–4k tokens
Standard modern LLMs	8k–16k tokens
Extended-context models	32k–128k+ tokens
Long-context research models	200k–1M+ tokens

Memory vs. Context Window in Agentic AI

- Understanding the difference between a context window and agent memory is essential for understanding how agentic AI achieves long-term, coherent behavior.
- The context window lets an LLM think; agent memory lets an AI agent remember.

Context Window

- **Temporary**: Exists only during the current interaction or reasoning step
- **Token-limited**: Strictly bounded by the model's maximum input size
- **Passive**: The model can only use what is explicitly provided in the prompt
- **Unstructured**: Plain text with no indexing or persistence

➞ Once the context window is exceeded or the session ends, the information is lost.

Agent Memory

- **Persistent**: Stored across steps, tasks, and sessions
- **Scalable**: Can grow using external storage (databases, vector stores, logs)
- **Actively retrieved**: Relevant memories are selected based on goals or state
- **Structured & indexed**: Organized as episodic, semantic, or procedural memory

➞ Enables learning from experience and continuity over time.

Multi Agent Systems

- Agentic AI systems can be designed with **one autonomous agent** or with **multiple interacting agents**. The choice affects scalability, intelligence, robustness, and system complexity.

A **multi-agent system** consists of **multiple autonomous agents** that:

- Have individual goals or roles
- Interact through communication, cooperation, or competition
- Coordinate to achieve system-level objectives

Characteristics

- Distributed intelligence
- Role specialization
- Communication and coordination mechanisms
- Can be cooperative, competitive, or hybrid

Strengths

- Parallel reasoning and execution
- Role specialization (planner, critic, executor, etc.)
- Greater scalability and robustness
- Better handling of complex, open-ended problems

Limitations

- Higher system complexity
- Communication and coordination costs
- Risk of conflicts, redundancy, or misalignment
- Harder evaluation and debugging

Examples

- Teams of LLM agents (planner + executor + reviewer)
- Simulated societies or markets
- Swarm robotics
- Multi-agent game environments

Aspect	Single-Agent	Multi-Agent
Intelligence	Centralized	Distributed
Scalability	Limited	High
Parallelism	Low	High
Coordination	Not needed	Essential
Robustness	Lower	Higher
Design complexity	Simpler	More complex

Agent architectures in agentic AI

Agent architectures define the **internal structure and control flow** of an AI agent—that is, **how the agent reasons, makes decisions, and takes actions** to achieve its goals.

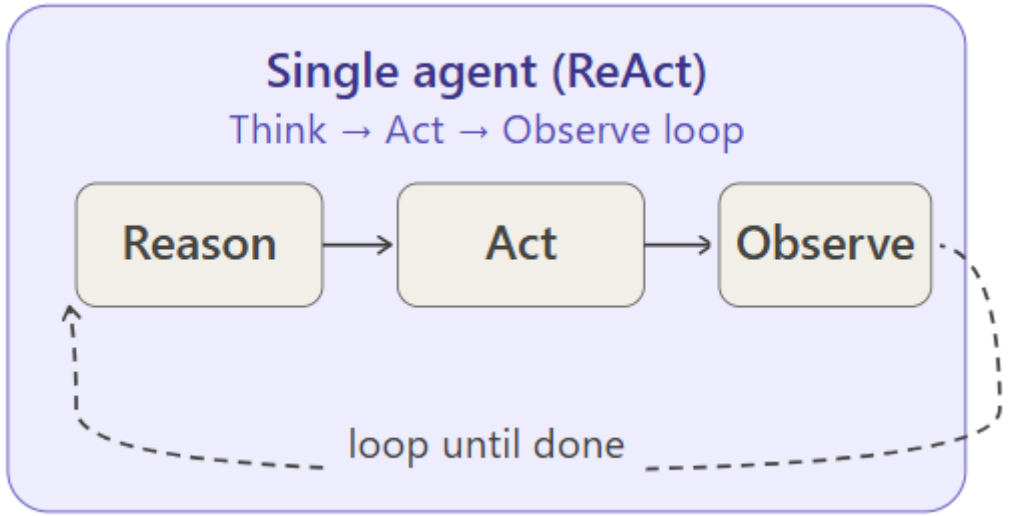
In practice, an agent architecture specifies:

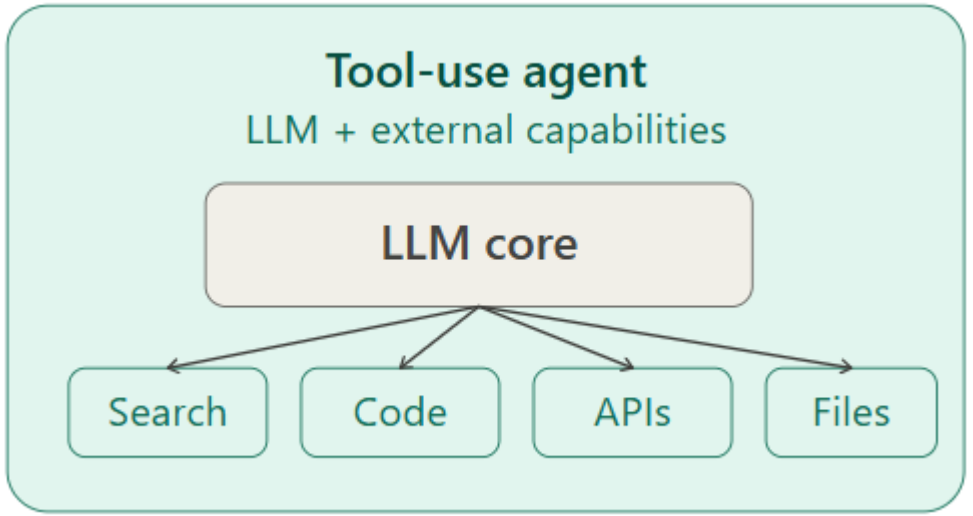
- **How reasoning is done** (reactive rules, deliberative planning, LLM-based reasoning)
- **How decisions are made** (step-by-step, plan-first, hierarchical)
- **How actions are executed** (direct actions, tool use, coordinated behaviors)
- **How information flows** between perception, memory, planning, and action

Agent architecture Types

- **Single Agent (ReAct)** is the foundational pattern — one LLM that cycles through Reasoning, Acting, and Observing until a task is complete. It's simple, debuggable, and surprisingly capable for contained tasks like answering questions with search or writing code.

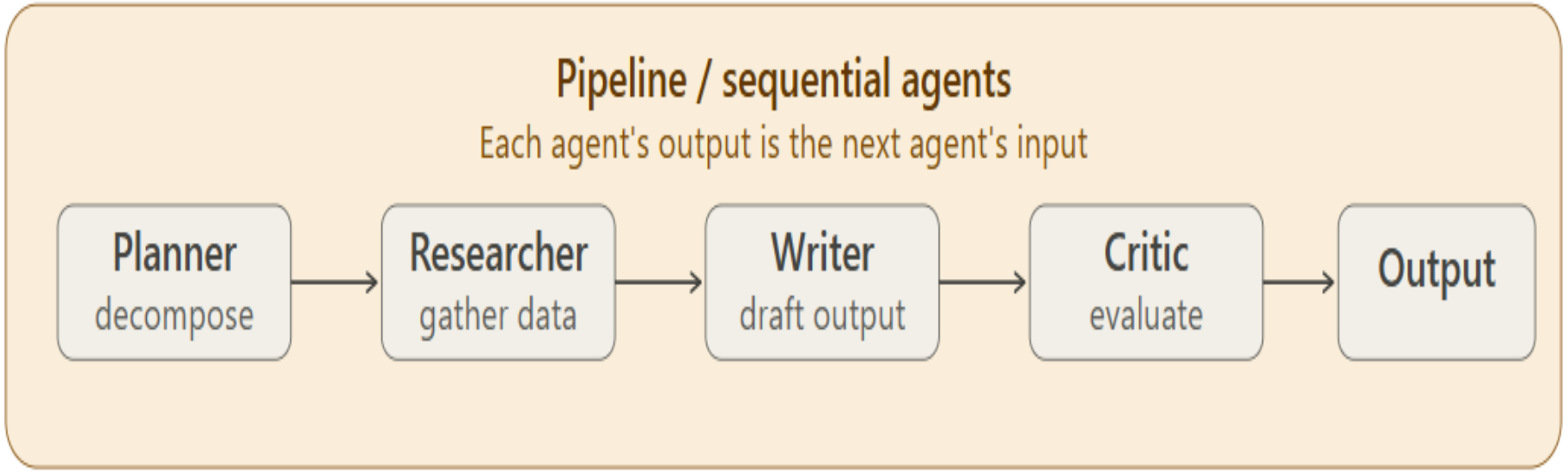
Tool-Use Agent extends a single agent by giving it access to external capabilities: web search, code execution, APIs, file systems. The LLM decides *when* and *which* tools to call. This is the most common production pattern today (think Claude with tools, or GPT with plugins).



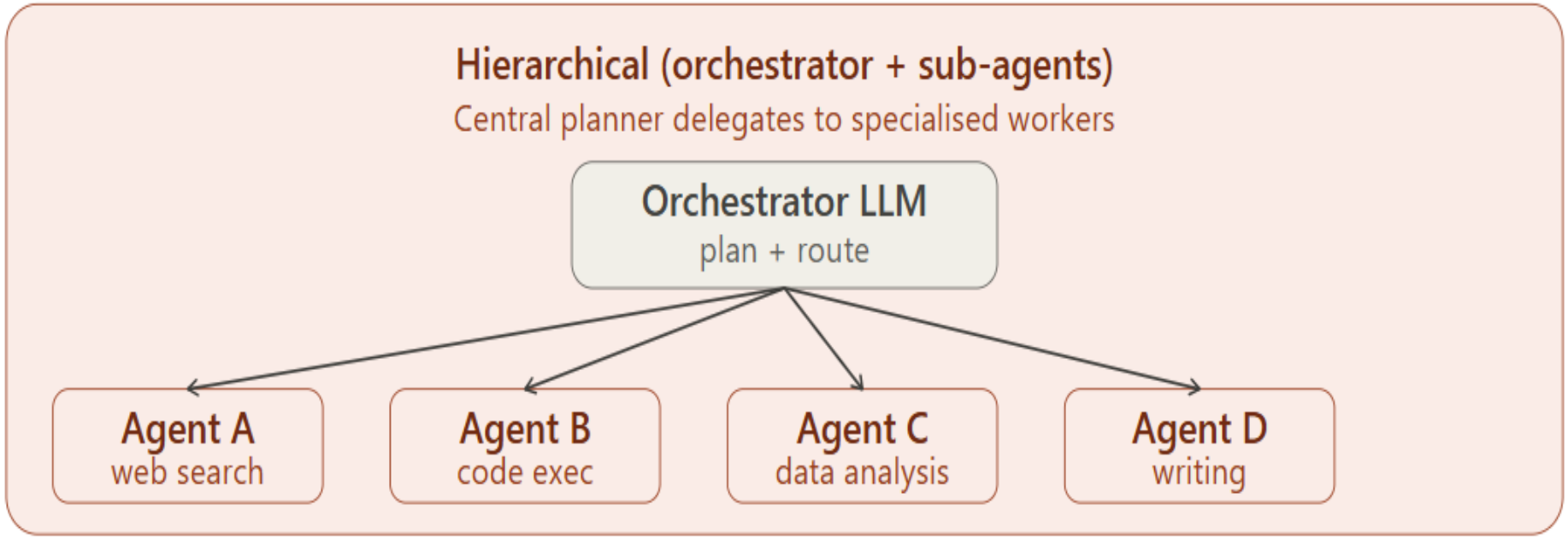


Agent architecture Types

Pipeline / Sequential arranges specialized agents in a fixed chain — a Planner breaks the task, a Researcher gathers information, a Writer drafts, a Critic evaluates. Each agent sees only the previous agent's output. This works well when the workflow is predictable and stages are clearly separable.

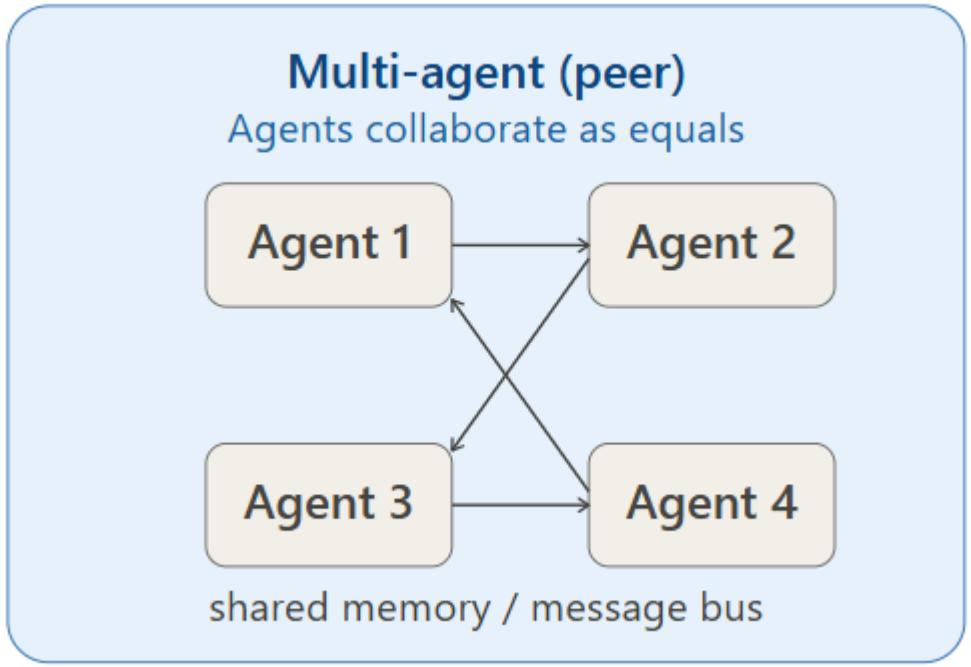


Hierarchical (Orchestrator + Sub-agents) has a central "manager" LLM that plans and routes work to specialized sub-agents running in parallel or series. The orchestrator synthesizes results. This is ideal for complex, multi-domain tasks — think "research + code + write a report" where each sub-agent can focus on its specialty.

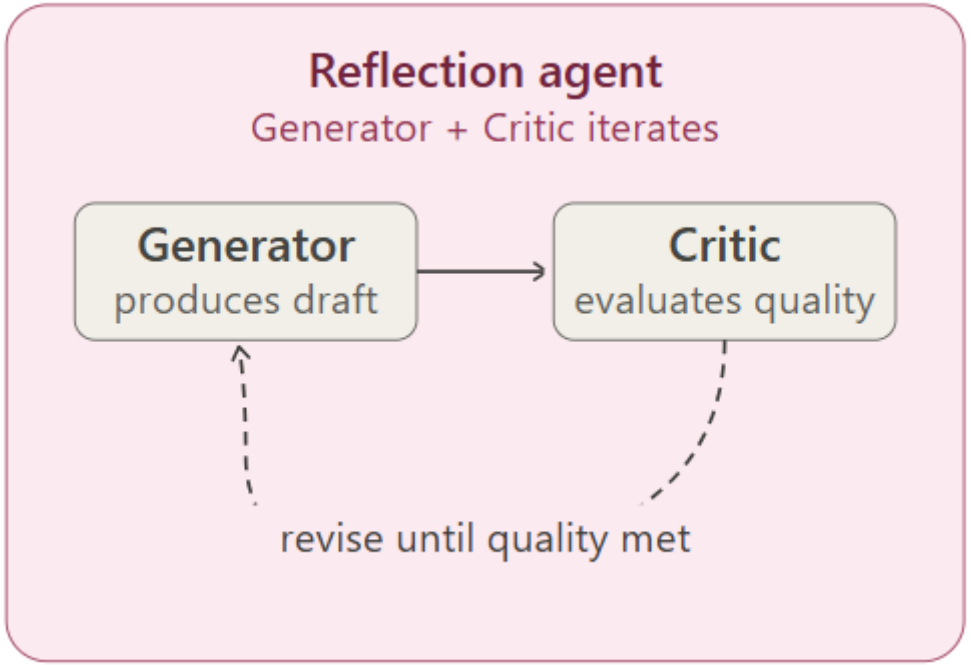


Agent architecture Types

Multi-Agent Peer Networks have agents communicating as equals over a shared message bus or memory. There's no fixed boss — agents can debate, vote, or specialize dynamically. This suits tasks benefiting from diverse perspectives (red-teaming, consensus-building, simulated societies).



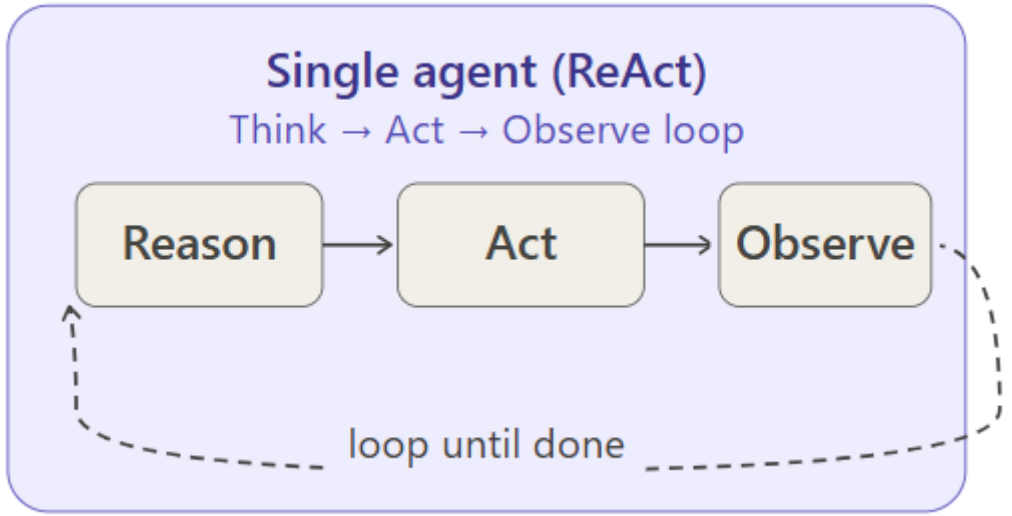
Reflection / Self-Critique pairs a Generator with a Critic in a loop. The generator produces a draft; the critic scores it and explains flaws; the generator revises. This dramatically improves output quality at the cost of more compute — widely used for code generation and long-form writing.

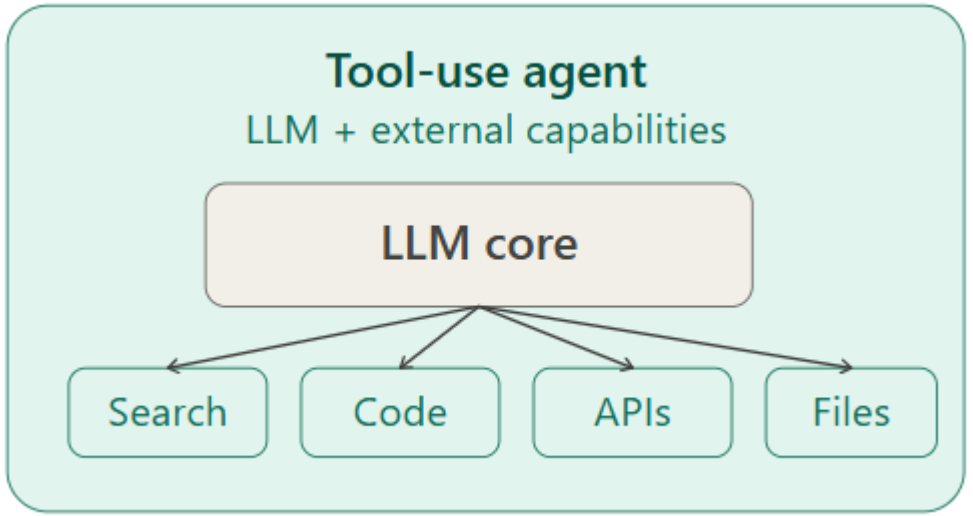


Agent architecture Types

- **Single Agent (ReAct)** is the foundational pattern — one LLM that cycles through Reasoning, Acting, and Observing until a task is complete. It's simple, debuggable, and surprisingly capable for contained tasks like answering questions with search or writing code.

Tool-Use Agent extends a single agent by giving it access to external capabilities: web search, code execution, APIs, file systems. The LLM decides *when* and *which* tools to call. This is the most common production pattern today (think Claude with tools, or GPT with plugins).





Agentic AI Use Cases

SOFTWARE ENGINEERING

Autonomous coding
Write, test, debug, and refactor code end-to-end from a single prompt

Explore ↗

CI/CD automation
Monitor builds, triage failures, open PRs, and run deployments autonomously

Explore ↗

Legacy migration
Translate codebases across languages or frameworks with verification loops

Explore ↗

KNOWLEDGE WORK & RESEARCH

Deep research
Multi-hop search, source synthesis, and long-form report generation

Explore ↗

Scientific literature review
Scan papers, extract findings, identify gaps, and surface hypotheses

Explore ↗

Competitive intelligence
Track rivals, parse filings, and surface strategic insights continuously

Explore ↗

Agentic AI Use Cases

CUSTOMER & BUSINESS OPERATIONS

Autonomous support
Resolve tickets end-to-end: lookup, action, escalate — without human relay

Explore ↗

Sales & CRM automation
Qualify leads, draft outreach, update CRM, and schedule follow-ups

Explore ↗

Finance workflows
Invoice processing, reconciliation, anomaly flagging, and report generation

Explore ↗

DATA & ANALYTICS

Autonomous data analysis
Write queries, execute, interpret, and iterate until the insight is found

Explore ↗

ETL & pipeline creation
Discover schemas, transform data, and build pipelines with validation

Explore ↗

BI & dashboard automation
Generate and refresh reports, charts, and KPI summaries on a schedule

Explore ↗

Agentic AI Use Cases

SECURITY & INFRASTRUCTURE

Threat detection & response
Correlate signals, investigate alerts, and contain incidents autonomously

Explore ↗

Vulnerability scanning
Scan code and infra, reproduce findings, and generate remediation PRs

Explore ↗

Infra provisioning
Spin up, configure, monitor, and scale cloud resources via IaC agents

Explore ↗

EMERGING & FRONTIER

Drug discovery
Generate hypotheses, design experiments, and interpret assay results

Explore ↗

Robotics & physical AI
Plan and execute multi-step manipulation tasks in physical environments

Explore ↗

Personal AI assistant
Manage email, calendar, tasks, and research as a proactive life-OS

Explore ↗

Thank You for Your Attention