# Agentic AI and Multi-Agent Systems: A Deep Conceptual and Systems-Level Understanding

Learnings from AutoDevX, IBM Agentic AI, and Google Agent Development Kit

Rujula Rahate

January 13, 2026

## Contents

# 1 Introduction: Why Agentic AI Exists

Modern AI systems began as function approximators: given an input, produce an output. Even large language models, despite their impressive reasoning abilities, fundamentally operate in this paradigm. They respond to prompts, but they do not *own* goals, they do not decide when to act, and they do not persist across time unless explicitly engineered to do so.

Agentic AI emerges from the realization that many real-world problems cannot be solved by single-step prediction. Instead, these problems require systems that can operate continuously, adapt to new information, and make decisions under uncertainty. Examples include software testing, autonomous debugging, workflow orchestration, robotics, and complex decision support systems.

An agentic system is therefore not defined by the presence of an LLM, but by the presence of **agency**: the ability to observe an environment, reason about it, act upon it, and update internal beliefs over time. The AutoDevX resources strongly emphasize this shift from "model-centric AI" to "system-centric AI".

# 2 From Reinforcement Learning to Agentic AI

Agentic AI is conceptually closest to reinforcement learning among all machine learning paradigms. In reinforcement learning, an agent interacts with an environment, takes actions, receives feedback, and updates its behavior based on long-term outcomes. Agentic AI generalizes this idea to environments that are not explicitly modeled as MDPs and where rewards may be implicit, sparse, or human-defined.

A key insight from studying both RL and agentic systems is that the *agent–environment loop* is the fundamental abstraction. Whether the environment is a simulator, a web application, an API ecosystem, or a human user, the same principles apply: partial observability, delayed consequences, and the need for exploration.

However, agentic AI differs from classical RL in one crucial way: instead of learning a numeric policy via gradient descent, agents often rely on structured reasoning, tool invocation, and memory. This makes them more interpretable, more flexible, and easier to align with human intent.

# 3 What Is an Agent? A Systems Definition

An agent is a persistent computational entity that maintains internal state and repeatedly executes a decision-making loop. Unlike a stateless model invocation, an agent exists across time and across interactions.

At a systems level, an agent consists of four tightly coupled components:

1. **Perception**: how the agent observes the environment
2. **Reasoning**: how the agent interprets observations and decides what to do
3. **Action**: how the agent affects the environment
4. **Memory**: how the agent retains and uses past information

Removing any one of these components collapses agency. Without memory, behavior is reactive. Without actions, the agent cannot influence outcomes. Without reasoning, behavior degenerates into scripts.

## 4 Autonomy: Degrees, Not a Binary

One of the most misunderstood aspects of agentic AI is autonomy. Autonomy is not an on/off switch; it exists on a spectrum.

At the lowest level, an agent may simply execute predefined steps under supervision. At higher levels, the agent decides which tools to use, when to use them, and how to recover from failure. At the highest levels, the agent sets sub-goals, revises plans, and decides when a task is complete.

The AutoDevX materials emphasize that *useful autonomy is constrained autonomy*. Fully unconstrained agents are brittle, unsafe, and difficult to debug. Practical agentic systems therefore include guardrails such as step limits, tool permissions, validation checks, and human approval points.

## 5 Reasoning and Planning in Agentic Systems

Reasoning in agentic systems is not equivalent to producing a correct answer. Instead, it is the process of deciding *what to do next*. This includes identifying missing information, choosing tools, and sequencing actions.

Planning is closely related but operates at a higher level. A plan is a provisional sequence of actions intended to achieve a goal. Importantly, plans in agentic systems are rarely fixed. They are continuously revised as new observations are made.

This dynamic planning capability is essential in open-ended environments such as software systems, where actions may fail, return unexpected results, or change the environment in unforeseen ways.

## 6 Memory: The Backbone of Agency

Memory transforms a reactive system into a learning system. In agentic AI, memory is not just a cache; it is a core design component.

Short-term memory holds the current conversational or task context. Long-term memory stores accumulated knowledge, past experiences, and learned heuristics. Episodic memory records sequences of interactions, allowing the agent to reason about cause and effect over time.

A critical insight from the learning resources is that memory design determines agent behavior. Poor memory design leads to repetition, confusion, and lack of progress. Good memory design enables adaptation, learning, and strategic behavior.

## 7 Tools as First-Class Citizens

Agentic systems do not operate in isolation. They act on the world through tools. A tool may be an API, a database, a compiler, a web browser, or a testing framework.

The Google Agent Development Kit treats tools as first-class abstractions. The agent reasons about which tool to use, what inputs to provide, and how to interpret outputs. Tool usage is therefore a decision-making problem, not a hardcoded pipeline.

This perspective is critical for building scalable agentic systems. Instead of embedding logic inside the agent, capabilities are exposed as tools, allowing agents to flexibly compose behaviors.

## 8 Multi-Agent Systems: When One Agent Is Not Enough

Single-agent systems are sufficient for many tasks, but they struggle when responsibilities grow complex. Multi-agent systems (MAS) address this by decomposing functionality across multiple

agents.

Each agent in a MAS typically has a well-defined role. For example, one agent may focus on planning, another on execution, and another on evaluation. This separation of concerns improves robustness and scalability.

However, MAS design introduces new challenges: coordination, communication overhead, and conflict resolution. The playbooks emphasize that MAS should be designed intentionally, not by simply running multiple agents in parallel.

## 9 Coordination, Communication, and Control

Agents in a multi-agent system must exchange information to function coherently. This communication can be explicit, through message passing, or implicit, through shared state.

Coordination mechanisms ensure that agents do not duplicate work, contradict each other, or enter deadlock. Control mechanisms define how decisions are escalated, merged, or overridden.

These concerns closely mirror distributed systems design, reinforcing the idea that agentic AI is fundamentally a systems engineering discipline.

## 10 Google Agent Development Kit: A Practical Framework

The Google Agent Development Kit provides a concrete instantiation of agentic AI principles. Rather than abstract theory, ADK focuses on how agents are built, orchestrated, and deployed in practice.

A key takeaway from studying ADK is the importance of explicit control flow. Agent execution is not left implicit or emergent; instead, it is structured, observable, and debuggable. This makes agentic applications maintainable and safe to deploy in production environments.

The provided samples demonstrate how complex behaviors emerge from simple agent loops combined with well-designed tools and memory.

## 11 Why This Matters for Real Applications

Agentic AI is not a research novelty; it is a response to the limitations of traditional AI systems. Tasks such as automated software testing, system monitoring, and adaptive workflows require systems that can operate continuously, reason about outcomes, and adapt strategies over time.

By combining reinforcement learning intuition with agentic system design, it becomes possible to build AI systems that are both flexible and controllable. This synthesis is what makes agentic AI a powerful paradigm for real-world applications.

## 12 Conclusion

Through studying the AutoDevX resources, IBM's Agentic AI guide, and Google's Agent Development Kit, I developed a deep systems-level understanding of agentic AI. Agents are not defined by models alone, but by the structure of interaction, reasoning, memory, and action.

Agentic AI represents a shift from building predictors to building decision-making systems. By grounding autonomy in structured reasoning, constrained execution, and thoughtful system design, agentic AI provides a practical path toward intelligent, adaptive, and scalable AI applications.