What Is a Multi-Agent System?

One agent is a worker. A multi-agent system is a team — several specialised agents that divide a task, coordinate through a defined structure, and hand work between them to reach an answer no single agent was built to give.

Alain AI Lab Research · Published July 14, 2026 · 10 min read

A **multi-agent system** is what you get when you stop asking one AI agent to do everything and instead give a problem to several agents that each own a piece of it. Where a single agent runs one loop — read the task, reason, use a tool, observe, repeat — a multi-agent system runs several such loops in parallel or in sequence and adds a coordination layer to decide who does what and how their outputs combine. If a single agent is one worker at a desk, a multi-agent system is a small firm: a manager, a few specialists, and rules for passing work between them. To see why this is a natural next step, it helps to first be clear on the surrounding structure — the subject of our report on what an agentic system is.

AT A GLANCE

DEFINITION	ACADEMIC ROOTS	CORE ADDED PART
Several interacting agents solving one problem together	**Distributed AI, 1980s–90s; predates LLMs**	**Coordination — who acts, when, how outputs merge**
COMMON PATTERNS	MAIN COST	HONEST RULE
Orchestrator–workers, hierarchical, sequential, network, group-chat	**Coordination tax — tokens, latency, error propagation**	**Start single; add agents only when one clearly fails**

01

More Than One Mind on the Problem

The clearest way to define a multi-agent system is by contrast. A single agent is a self-contained loop wrapped around a language model: it perceives a task, reasons, acts through a tool, observes the result, and repeats until done. That cycle is the whole engine, and the

report on how AI agents work walks through it in detail. A multi-agent system keeps that engine but runs several copies of it, each pointed at a different slice of the problem, and adds a concern the single agent never had: coordination.

That second concern is the entire subject. Beyond what each agent should do, you must answer who decides, how agents talk to each other, in what order they act, and how their separate outputs get stitched back into one result. A multi-agent system is therefore not simply "more agents." It is more agents plus the machinery that keeps them working toward one goal. Take that machinery away and you do not have a team — you have a crowd.

02

An Idea Older Than the Language Model

It is tempting to treat multi-agent systems as a product of the recent AI boom, but the idea is decades old. The field of distributed artificial intelligence explored how autonomous programs could cooperate, compete, and negotiate throughout the 1980s and 1990s, long before large language models existed. Michael Wooldridge's textbook *An Introduction to MultiAgent Systems*, first published in 2002, codified the vocabulary — agents, environments, coordination, negotiation — that the field still borrows from today.

What changed recently is not the concept but the parts. The modern version uses a language model as a general-purpose reasoning core, so an "agent" can now be defined largely by a prompt, a role, and a set of tools rather than bespoke logic. That lowered the cost of creating a specialised agent to almost nothing, which is why the pattern suddenly appears everywhere. But the older literature already mapped the hard problems — coordination overhead, conflicting goals, communication cost — that today's builders keep rediscovering.

03

Why Split the Work at All

If one capable agent can loop through a task, why bother with several? The honest answer is a handful of concrete advantages, not a belief that more is better. The first is **specialisation**. A single prompt asked to research, write, fact-check, and format tends to do each job adequately and none well. Splitting those into separate agents — each with a focused instruction and only the tools it needs — often raises the quality of each step, the way a newsroom separates reporter, editor, and fact-checker.

The second is **modularity**: when each agent owns a narrow job, you can test, swap, or improve one without rewriting the whole system, and a failure is easier to locate. The third is

parallelism — independent sub-tasks can run at the same time rather than one after another, cutting wall-clock time on breadth-heavy work, while each agent carries only the context its slice requires. These are real gains, but each is bought with coordination cost, which is why they only pay off when the task is genuinely large or mixed.

> A single agent asked to research, write, and fact-check does all three passably. A multi-agent system gives each job its own specialist — but now something has to manage them, and that manager is where multi-agent design lives or dies.

04

How Agents Are Arranged: The Coordination Patterns

Most multi-agent systems fall into a small number of arrangements. These patterns overlap and the names are not perfectly standardised, so treat them as a common vocabulary rather than a fixed taxonomy. The most widely used is the **orchestrator–workers** pattern: a lead agent breaks the problem into sub-tasks, hands each to a worker, and synthesises their results. Anthropic's December 2024 essay "Building Effective Agents" describes exactly this shape, a central model delegating to and then combining the work of others.

A close relative is the **hierarchical** arrangement, where orchestration nests — a top agent directs mid-level agents that each direct their own workers, like an org chart. The **sequential** or pipeline pattern is simpler: agents form an assembly line, each taking the previous one's output as its input, useful when a task has clear ordered stages. The **network** or peer-to-peer pattern lets agents talk to one another freely without a single boss, trading control for flexibility. Finally, the **group-chat** or debate pattern puts several agents in a shared conversation where they propose, critique, and refine. Which arrangement fits depends on the task: ordered work wants a pipeline, breadth-heavy work wants orchestrator–workers.

05

How Agents Actually Talk to Each Other

An arrangement is only a diagram until agents can pass information, which they do in two main ways. The first is **message passing**: one agent's output becomes the text another reads as input. The second is **shared state**: agents read from and write to a common store — a scratchpad, a task list, a structured object — so each can see the state of the work without being handed it directly. Most real systems mix both.

The frameworks that build these systems each lean on a different metaphor, a subject covered in our comparison of the major agent frameworks. Microsoft's AutoGen grew up around conversable agents that exchange messages in a group chat, with an option to insert a human into the loop. CrewAI organises agents into role-based "crews" that run through a defined process, sequential or manager-led. LangGraph models the system as a graph, where nodes are agents and edges are the control flow, allowing loops and branching. A recurring device across all of them is the **supervisor** or router: a coordinating agent whose only job is to decide which agent acts next based on the current state — often the real brain of a multi-agent system.

06

When Many Agents Beat One — and When They Don't

The honest position is that multi-agent systems are not automatically better; they are better for a specific kind of problem. Anthropic's June 2025 write-up on its own multi-agent research system is instructive. An orchestrator directing several parallel subagents outperformed a single agent on broad, open-ended research that benefits from exploring many directions at once — but it consumed roughly fifteen times the tokens of a plain chat. That figure is theirs, tied to their setup, and the lesson generalises even if the exact multiple does not: coordination and parallel exploration cost real money and real latency.

This is why experienced builders follow a single-agent-first rule. Start with the simplest thing that could work — one agent, one loop — and reach for multiple agents only when a single one demonstrably fails: when the task is too broad for one context window, when independent sub-tasks can run in parallel, or when distinct specialisations clearly raise quality. Reaching for a multi-agent architecture because it sounds sophisticated is how projects end up slow and expensive for a result a single well-prompted agent could have produced.

07

Multi-Agent Systems in Crypto

In crypto the multi-agent pattern maps naturally onto division of labour around money. A common design splits responsibilities: a **watcher** agent monitors on-chain data and prices, a **risk** agent evaluates whether a proposed action is within limits, and an **executor** agent assembles and submits the transaction. Separating these mirrors how a disciplined trading desk keeps analysis, risk control, and execution in different hands so no single point makes an unchecked decision.

But the same split multiplies the security surface. Every agent that can touch funds is another component that needs a key, another place a hostile instruction could land, another handoff where context can be lost. Because a confirmed on-chain transaction is final — there is no support desk to reverse it — multi-agent crypto systems need constrained execution rails more than single-agent ones, not less: account-abstraction wallets under ERC-4337, session keys that grant each agent only narrow, time-boxed permissions, and multi-party computation that splits signing authority so no one agent holds an unrestricted key. The architecture makes the system capable; the rails make it safe to point at real balances.

08

The Coordination Tax and the Failure Modes

Every advantage of a multi-agent system is paid for with what is fairly called a coordination tax. The first line item is **cost and latency**: more agents mean more model calls, more tokens, and more waiting, as Anthropic's roughly fifteen-times figure makes concrete. The second is **error propagation**: a mistake made early by one agent is passed downstream and amplified, because later agents trust their input. The third is **context loss across handoffs**: information clear inside one agent can be dropped or garbled when summarised for the next, so the system as a whole knows less than any single agent did.

The fourth is **coordination failure** itself — agents waiting on each other, looping without progress, or disagreeing with no mechanism to resolve it. And the security picture is amplified, not merely additive. The OWASP security project ranks **prompt injection** as the top risk to language-model applications in its 2025 list, and a multi-agent system has more surfaces where hostile text can enter and more handoffs where it can spread. OWASP's companion warning about **excessive agency** — granting a system more autonomy than its reliability earns — lands harder when several semi-autonomous agents act at once. The early autonomous projects AutoGPT and BabyAGI, both from 2023, showed how quickly loosely-coordinated agent loops get stuck or burn cost without finishing. The tooling is far better now, but the blunt question survives every architecture: what is the worst thing this system of agents can do before a human is asked?

"Two are better than one; because they have a good reward for their labour."

ECCLESIASTES 4:9

METHODOLOGY & SOURCES

This report was produced by a parallel multi-agent research process, with each claim independently checked and labelled verified, partial, or uncertain before inclusion. Figures are stated as approximate or attributed to their source where precise public data does not exist; no specific decimals, dates, or dollar figures were invented, and unverified “nine-figure AI-agent hack” claims were excluded as unsupported.

Primary references include Michael Wooldridge’s *An Introduction to MultiAgent Systems* (2002) and the distributed-AI literature of the 1980s–90s; Anthropic’s “Building Effective Agents” (December 19, 2024) on orchestrator–workers, and its multi-agent research-system post (June 2025) reporting roughly 15× the token cost of a single chat; the coordination models of Microsoft’s AutoGen, CrewAI, and LangGraph; Gartner’s October 21, 2024 naming of agentic AI as a top trend; the OWASP Top 10 for LLM Applications (2025) on prompt injection and excessive agency; and the ERC-4337 account-abstraction standard. Educational content only; not financial advice.

Related reading: What Is an Agentic System? · AI Agent Frameworks: LangGraph, CrewAI & AutoGen · How Do AI Agents Work?

Alain AI Lab — independent crypto & AI research.
intelligencecrypto.org · This document is for educational purposes only and is not financial advice.