International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 5, Sep-Oct-2026, ISSN (Online): 2395-566X

Multi-Agent AI Systems: A Review of Architectures, Coordination Strategies and Challenges in Llm-Based Agent Collaboration

Dhole Om Santosh[1] Prof. N. G. Rokade[2] Prof. M. C. Kolse[3] Prof. S. K. Said4
Department of Artificial Intelligence and Data Science
Jaihind College of Engineering
Pune, Maharashtra, India

Abstract—Artificial Intelligence is moving from single models that answer one question at a time toward agentic systems that plan, use tools, coordinate subtasks and execute multi-step workflows. A Multi-Agent AI System (MAAS) contains two or more autonomous or semi-autonomous agents that interact to achieve a shared or related goal, where each agent may have a specialized role, memory, tools and decision process. This paper reviews the foundations and recent advances of such systems. It traces the classical theory of multi-agent systems, including agent communication languages and the contract-net protocol, and examines how large language models have revived the field through frameworks for role-playing, conversation-driven collaboration, standardized operating procedures and debate. Coordination topologies such as orchestrator–worker, hierarchical, peer-to-peer and blackboard designs are compared, along with mechanisms for memory, tool access and verification. A layered reference architecture is proposed that separates user interface, orchestration, specialized agents, shared memory and tools, and governance. The paper also analyses failure modes, including error propagation, coordination overhead, prompt injection and cost, and identifies research gaps in evaluation and reliability. It concludes that multi-agent designs are most valuable when tasks decompose naturally and when intermediate outputs are verified.

Index Terms—Multi-agent systems, large language models, AI agents, agent orchestration, tool use, coordination, agent communication, autonomous systems

I. INTRODUCTION

Most early applications of large language models (LLMs) followed a simple pattern: a user asks a question and the model produces an answer. Many real tasks, however, require several steps, different kinds of expertise and interaction with external tools. Writing software, researching a topic or analysing a business problem involves planning, gathering information, producing drafts and checking them.

Multi-agent systems have long been studied in AI. Wooldridge [1] defines an agent as a computer system situated in an environment and capable of autonomous action to meet its objectives, and a multi-agent system as one in which several such agents interact. The arrival of capable LLMs has given these ideas new life: an LLM prompted with a role, tools and memory can act as a flexible agent, and several such agents can collaborate.

This paper reviews multi-agent AI systems with a focus on LLM-based agents. It surveys classical and modern work, proposes a reference architecture, compares coordination patterns and discusses open challenges.

II. BACKGROUND AND RELATED WORK

2.1 Components Of An Agent

An LLM-based agent typically consists of a profile (role and instructions), memory (short-term context and long-term storage), planning (decomposing goals into steps) and action (calling tools or messaging other agents) [2]. ReAct [3] showed that interleaving reasoning traces with actions improves task success and transparency.

2.2 Coordination Topologies

Agents can be organized in several ways, as illustrated in Fig. 1. In an orchestrator–worker design, a lead agent assigns subtasks to specialists. In a hierarchy, managers supervise teams. In peer-to-peer designs, agents converse as equals, and in a blackboard design, agents read and write to a shared workspace.

© 2026 IJSRET

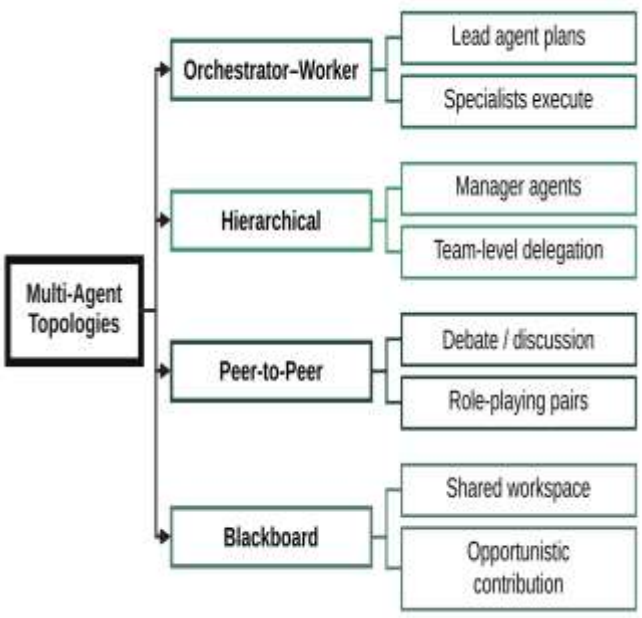


Fig. 1. Coordination topologies in multi-agent AI systems.

2.3 Review Of Related Studies

Classical research established the vocabulary of agent cooperation. Smith's contract-net protocol [4] introduced task allocation through announcements, bids and awards, and the FIPA agent communication language [5] standardized message types such as request, inform and propose. In reinforcement learning, Lowe *et al.* [6] proposed centralized training with decentralized execution for cooperative and competitive agents, and Vinyals *et al.* [7] used multi-agent self-play to reach grandmaster level in StarCraft II.

With LLMs, Park *et al.* [8] created generative agents that simulate believable human behaviour in a small town using memory streams, reflection and planning. CAMEL [9] explored role-playing in which a user agent and an assistant agent cooperate on a task with minimal human input. AutoGen [10] provided a general framework in which customizable agents converse to solve tasks, combining LLMs, tools and human feedback.

Structured collaboration improves reliability. MetaGPT [11] encoded standardized operating procedures into roles such as product manager, architect and engineer, reducing errors that arise from unstructured chatting. ChatDev [12] organized agents into a virtual software company that designs, codes and tests programs. Du *et al.* [13] showed that multiple models debating their answers over several rounds improves factual accuracy and reasoning, and Reflexion [14] used verbal self-feedback to improve performance across attempts.

Surveys by Xi *et al.* [15] and Guo *et al.* [16] classify these systems by profile, communication and capability acquisition, and note that evaluation and safety remain immature. Greshake *et al.* [17] demonstrated indirect prompt injection, in which content retrieved by an agent contains hidden instructions, a risk that multiplies when agents pass outputs to one another. Table I summarizes the surveyed work.

TABLE I Summary of Literature Survey

Sr. No.	Ref.	Year	Methodology	Findings	Research Gap
1	[4]	1980	Contract-net protocol	Task allocation via announce–bid–award	Assumes reliable, honest agents
2	[6]	2017	Multi-agent actor-critic (MADDPG)	Centralized training, decentralized execution	Reinforcement learning setting only
3	[8]	2023	Generative agents with memory and reflection	Believable simulated human behaviour	High cost; simulation focus
4	[9]	2023	CAMEL role-playing agents	Autonomous cooperation with minimal prompting	Role drift and loops
5	[10]	2023	AutoGen conversation framework	Flexible multi-agent patterns with tools	Framework; reliability depends on design
6	[17]	2023	Indirect prompt injection	Showed attacks via retrieved content	Defences remain incomplete
7	[11]	2024	MetaGPT standardized procedures	Fewer errors via structured roles	Mainly software tasks
8	[12]	2024	ChatDev virtual software company	End-to-end software generation	Generated code needs testing
9	[13]	2024	Multi-agent debate	Improved factuality and reasoning	Multiplies inference cost

© 2026 IJSRET

III. RESEARCH GAP AND OBJECTIVES

3.1 Identified Research Gaps

The following gaps are identified:

1. Errors made by one agent can propagate and be amplified by others, and systematic verification is often missing.

2. Benchmarks rarely measure cost, latency and robustness alongside task success.

3. Guidance on when a multi-agent design outperforms a single well-prompted agent is limited.

4. Security of inter-agent communication and tool access, especially against prompt injection, is weakly addressed.

5. Standard interfaces for tools and agent-to-agent messaging are still emerging.

3.2 Objectives Of The Review

The objectives of this review are:

- To study the components and coordination topologies of multi-agent AI systems.
- To review classical and LLM-based multi-agent frameworks and their contributions.
- To propose a layered reference architecture with verification and governance.
- To compare coordination patterns in terms of reliability, cost and scalability.
- To identify challenges and research directions for trustworthy multi-agent systems.

I. IV. Design of the Proposed System

Fig. 2 presents the proposed architecture. It follows an orchestrator–worker pattern with explicit verification, because this structure is easy to reason about and to audit.

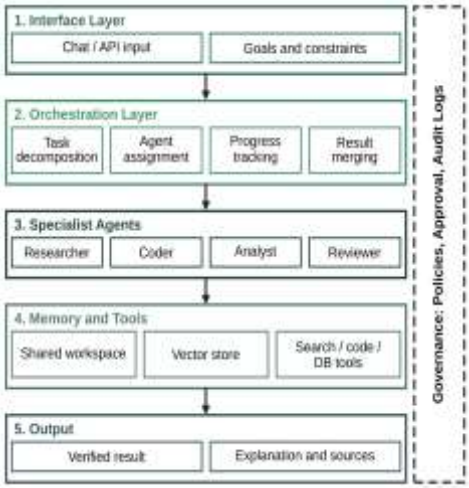


Fig. 2. Proposed layered reference architecture for a multi-agent AI system.

3.3 Interface Layer

Users submit goals through a chat interface or an API. The interface collects constraints such as budget, deadline and required output format.

3.4 Orchestration Layer

The orchestrator decomposes the goal into subtasks, assigns them to specialist agents, tracks progress and merges results. It can run independent subtasks in parallel and re-plan when a subtask fails.

3.5 Specialist Agent Layer

Each agent has a role, such as researcher, coder, analyst or writer, with its own instructions and permitted tools. A reviewer agent checks intermediate outputs against requirements before they are merged.

3.6 Memory And Tools Layer

A shared workspace stores intermediate artefacts, and a vector store provides long-term retrieval [18]. Tools such as web search, code execution and databases are exposed through standardized interfaces such as the Model Context Protocol [19].

3.7 Governance Layer

Policies restrict which tools each agent may use, sensitive actions require human approval, and all messages and tool calls are logged for audit.

3.8 Methodology

Fig. 3 shows the execution cycle. If a task is split into k subtasks, each completed correctly with probability p_i, and results are accepted only after review, the probability that the final output is correct without verification is approximately

$$P(success) \approx p_1 \cdot p_2 \cdot \ldots \cdot p_k \qquad (1)$$

which falls quickly as k grows. This simple relation explains why verification steps and small numbers of well-defined subtasks are essential.

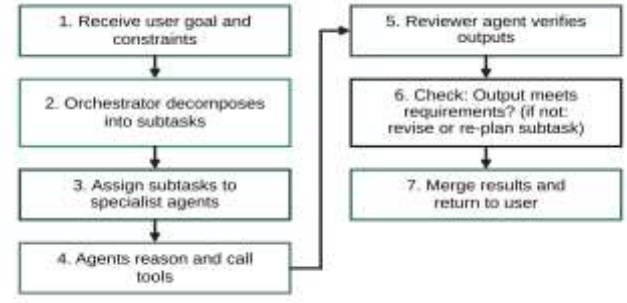


Fig. 3. Execution cycle of an orchestrated multi-agent system.

© 2026 IJSRET

Evaluation should record task success rate, number of model calls, total token cost, wall-clock time and the frequency of human intervention.

IV. DISCUSSION AND COMPARISON

Table II compares coordination patterns. Orchestrator–worker designs are predictable and efficient; hierarchies scale to large tasks but add overhead; peer debate improves accuracy at higher cost; and blackboard designs support flexible collaboration but require careful conflict handling.

Pattern	Reliability	Cost	Scalability
Single agent	Medium	Low	Low
Orchestrator–worker	High (with review)	Medium	Medium–High
Hierarchical	Medium–High	High	High
Peer debate	High (accuracy)	High	Low–Medium
Blackboard	Medium	Medium	Medium

TABLE II Comparison of Coordination Patterns

V. CHALLENGES AND FUTURE DIRECTIONS

Table III summarizes the main challenges. Promising directions include learned orchestration policies that decide when to spawn agents, formal verification of agent plans, standardized agent-to-agent protocols, smaller specialized models to reduce cost, and benchmarks that reflect real workflows. Analyses of why multi-agent LLM systems fail will help designers avoid common pitfalls.

Challenge	Mitigation
Error propagation	Reviewer agents; tests; human checkpoints
Coordination overhead	Few, well-defined roles; parallel subtasks
Prompt injection	Tool permissions; content isolation
Cost and latency	Smaller models for simple roles; caching
Evaluation	Task-level benchmarks with cost metrics

TABLE III Challenges and Mitigation

VI. CONCLUSIONS

Multi-agent AI systems extend the capabilities of language models by dividing complex work among specialized agents that communicate, use tools and check one another. The reviewed literature shows clear benefits in software development, research and reasoning tasks, particularly when collaboration is structured by roles and procedures. Reliability, however, depends on verification, governance and secure tool access, because errors and attacks can spread between agents. Multi-agent designs are most effective when a task naturally decomposes into parts and when each part's output can be checked.

6.1 Acknowledgment

The authors express their sincere gratitude to the Department of Artificial Intelligence and Data Science, Jaihind College of Engineering, Pune, for providing the facilities, guidance and encouragement needed to carry out this work. The authors also thank the faculty members and colleagues whose valuable suggestions helped improve the quality of this review.

REFERENCES

1. M. Wooldridge, An Introduction to MultiAgent Systems, 2nd ed. Chichester, U.K.: Wiley, 2009.
2. L. Wang et al., “A survey on large language model based autonomous agents,” Front. Comput. Sci., vol. 18, no. 6, Art. no. 186345, 2024.
3. S. Yao et al., “ReAct: Synergizing reasoning and acting in language models,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2023.
4. R. G. Smith, “The contract net protocol: High-level communication and control in a distributed problem solver,” IEEE Trans. Comput., vol. C-29, no. 12, pp. 1104–1113, 1980.
5. Foundation for Intelligent Physical Agents, “FIPA ACL message structure specification,” Doc. SC00061G, 2002.
6. R. Lowe et al., “Multi-agent actor-critic for mixed cooperative-competitive environments,” in Proc. NeurIPS, vol. 30, 2017.
7. O. Vinyals et al., “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” Nature, vol. 575, pp. 350–354, 2019.
8. J. S. Park et al., “Generative agents: Interactive simulacra of human behavior,” in Proc. 36th ACM Symp. User Interface Softw. Technol. (UIST), 2023.
9. G. Li et al., “CAMEL: Communicative agents for “mind” exploration of large language model society,” in Proc. NeurIPS, vol. 36, 2023.
10. Q. Wu et al., “AutoGen: Enabling next-gen LLM applications via multi-agent conversation,” arXiv:2308.08155, 2023.
11. S. Hong et al., “MetaGPT: Meta programming for a multi-agent collaborative framework,” in Proc. ICLR, 2024.

© 2026 IJSRET

12. C. Qian et al., “ChatDev: Communicative agents for software development,” in Proc. 62nd Annu. Meeting Assoc. Comput. Linguist. (ACL), 2024.
13. Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, “Improving factuality and reasoning in language models through multiagent debate,” in Proc. ICML, 2024.
14. N. Shinn et al., “Reflexion: Language agents with verbal reinforcement learning,” in Proc. NeurIPS, vol. 36, 2023.
15. Z. Xi et al., “The rise and potential of large language model based agents: A survey,” arXiv:2309.07864, 2023.
16. T. Guo et al., “Large language model based multi-agents: A survey of progress and challenges,” in Proc. 33rd Int. Joint Conf. Artif. Intell. (IJCAI), 2024.
17. K. Greshake et al., “Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection,” in Proc. 16th ACM Workshop Artif. Intell. Security (AISec), 2023.
18. P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in Proc. NeurIPS, vol. 33, 2020, pp. 9459–9474.
19. Anthropic, “Model Context Protocol specification,” 2024. [Online]. Available: https://modelcontextprotocol.io

© 2026 IJSRET