

# The Orchestration of Multi-Agent Systems: Architectures, Protocols, and Enterprise Adoption

Apoorva Adimulam

*Applied Agentic AI*

*Skan AI*

Menlo Park, USA

apoorva.adimulam@skan.ai

Rajesh Gupta

*Applied Agentic AI*

*Skan AI*

Menlo Park, USA

rajesh.gupta@skan.ai

Sumit Kumar

*Applied Agentic AI*

*Skan AI*

Bengaluru, India

sumit.kumar@skan.ai

**Abstract**—Orchestrated multi-agent systems represent the next stage in the evolution of artificial intelligence, where autonomous agents collaborate through structured coordination and communication to achieve complex, shared objectives. This paper consolidates and formalizes the technical composition of such systems, presenting a unified architectural framework that integrates planning, policy enforcement, state management, and quality operations into a coherent orchestration layer. Another primary contribution of this work is the in-depth technical delineation of two complementary communication protocols—the Model Context Protocol, which standardizes how agents access external tools and contextual data, and the Agent-to-Agent protocol, which governs peer coordination, negotiation, and delegation. Together, these protocols establish an interoperable communication substrate that enables scalable, auditable, and policy-compliant reasoning across distributed agent collectives. Beyond protocol design, the paper details how orchestration logic, governance frameworks, and observability mechanisms collectively sustain system coherence, transparency, and accountability. By synthesizing these elements into a cohesive technical blueprint, this paper provides comprehensive treatments of orchestrated multi-agent systems—bridging conceptual architectures with implementation-ready design principles for enterprise-scale AI ecosystems.

**Index Terms**—Agent orchestration, Agent-to-Agent protocol (A2A), dynamic task allocation, Model Context Protocol (MCP), multi-agent systems, observability, state management, system governance

## I. INTRODUCTION

The landscape of LLM-powered agents has undergone a marked transformation. Early deployments prioritized isolated, task-specific agents, highly specialized systems with narrow operating scopes. However, contemporary trends point toward ecosystems of collaborating agents. This transition mirrors broader developments in distributed computing, where value emerges less from individual capabilities and more from orchestrated interactions within a collective.

Several technical drivers explain why the pivot to multi-agent architectures is emerging now including:

- scalability limits of LLMs, where context length and reasoning bottlenecks constrain performance
- the need for specialization versus generalization, enabling modular agents optimized for specific domains to be composed dynamically

- advances in communication protocols, with message-passing abstractions and nascent standards for inter-agent APIs
- economic efficiency, as distributed collectives of smaller agents often outperform costly all-purpose deployments as demonstrated in [1], [2]

Recent industry signals underscore the momentum of this transition. At the enterprise level, PwC has positioned its Agent OS [3] as a switchboard for multi-agent coordination, emphasizing composability and interoperability across enterprise functions. Similarly, Accenture's Trusted Agent Huddle [4] introduces governance mechanisms for secure, cross-organizational workflows and aligns with the emerging Agent-to-Agent (A2A) protocol. At the same time, research and development within the technical ecosystem has accelerated with frameworks such as LangChain, AutoGen, IBM Watsonx Orchestrate, and Google's Agent Development Kit providing modular infrastructure for coordination, negotiation, and role-based task delegation. Collectively, these initiatives signal rapid movement toward standardization and operational readiness.

The remainder of this paper progresses from conceptual foundations to practical realization. Section II traces the evolution of agentic systems toward orchestrated collectives, while Section III establishes the architectural composition of multi-agent systems and outlines their essential components. Sections IV–VII expand on these components in depth, examining how specialized agent roles, orchestration logic, communication protocols, and governance mechanisms together form the technical backbone of orchestrated intelligence. Section VIII presents real-world case studies that demonstrate the practical impact of these architectures across industries. Section IX discusses the challenges and risks associated with scaling multi-agent systems, and the future research directions aimed at improving efficiency, reliability, and interoperability. Finally, Section X concludes with key insights on how orchestrated multi-agent systems can serve as a foundation for enterprise-scale AI ecosystems.

## II. EVOLUTION OF AGENTIC SYSTEMS

Agentic systems are founded on the principle of autonomous entities that can perceive their environment, make decisions,

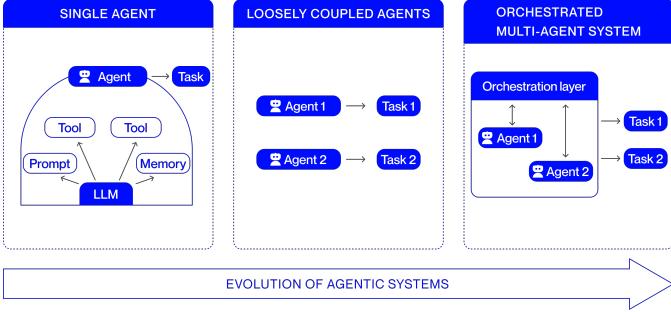


Fig. 1. Evolution of Agentic Systems

and take actions to achieve specific goals. Defined by autonomy, reactivity, proactivity, and social ability, they extend beyond scripted automation to operate adaptively.

Early deployments relied on single agents. These monolithic systems, with no coordination overhead, were well-suited to narrow tasks such as customer support chatbots that resolve FAQs, financial bots generating daily reports, or personal productivity assistants handling email and calendar management. Their reliability made them effective in bounded contexts, but they lacked scalability and adaptability for complex or dynamic environments.

To address these limitations, research and practice shifted toward loosely coupled agentic systems. In such systems, multiple agents operate in parallel with relative independence, and minimal interaction. This architecture enables specialization and the emergence of collective behaviors that a single agent cannot achieve. Recent examples include scientific research assistants [5] where literature-retrieval, reasoning, and validation agents collaborate to accelerate discovery; collaborative AI coding environments [6] where distinct agents write, review, and test code; news pipelines [7] where aggregation, fact-checking, and synthesis are distributed across agents; and autonomous driving ecosystems [8] where perception, navigation, and coordination agents cooperate loosely to ensure safe operation. Fig. 1 represents the discussed evolution of agentic systems.

### III. ARCHITECTURAL COMPOSITION OF MULTI-AGENT SYSTEMS

Building an orchestrated multi-agent system (MAS) involves more than simply connecting multiple autonomous agents. It requires designing specialized roles, establishing a coordination layer that governs their interactions, and defining clear communication protocols that allow agents to exchange information effectively.

Specialized agents operate as autonomous components that execute well-defined tasks within the system, each contributing a distinct capability toward the collective objective. Their interactions are coordinated through an orchestration layer that determines execution order, manages dependencies, and aligns individual outputs into a coherent operational flow. The exchanges rely on communication protocols that standardize how information is represented and transferred across agents,

ensuring semantic consistency and enabling the orchestration layer to maintain synchronized and interpretable system behavior. When these foundational components are applied in practice, they collectively enable complex, domain-specific workflows that require coordinated intelligence across multiple decision points.

The following sections examine these technical elements in greater depth, focusing on agent specialization, orchestration mechanisms, and communication strategies as the foundational components of multi-agent architectures.

### IV. SPECIALIZED AGENTS

In a multi-agent system, specialized agents are autonomous components designed to perform narrowly scoped, role-specific tasks within the broader architecture. Each agent typically incorporates a large language model as its cognitive core, enabling it to perceive inputs, reason about them, and act within clearly defined operational boundaries. By assigning distinct roles such as retrieval, reasoning, validation, or monitoring, the system decomposes complex objectives into smaller, coordinated subtasks. This division of labor promotes modularity and collaboration, allowing agents to complement one another's capabilities, reduce redundancy, and achieve outcomes that surpass those of a single, general-purpose agent. Through such specialization, a multi-agent system attains higher precision, scalability, and robustness while preserving clarity of function and accountability across its components.

Below are the key categories of specialized agents:

- **Worker Agents** - Worker agents represent the most basic but essential type. They are responsible for carrying out well-defined tasks such as a Retrieval-augmented generation (RAG) pipeline. In practice, some worker agents are stateless, handling each request independently without retaining context, while others are stateful, tracking progress across multiple steps in a workflow. In large systems, worker agents often operate in parallel, with each specializing in a narrow sub-domain. In a financial underwriting workflow, individual worker agents may extract applicant data from loan documents, compute preliminary credit scores, or generate draft risk assessments that downstream agents validate and consolidate. These agents form the execution layer of the system, performing domain-specific computations that feed subsequent validation and oversight processes.
- **Service Agents** - Service agents provide shared operational capabilities that other agents depend on during workflow execution. They act as reusable utilities within the multi-agent ecosystem, performing tasks such as quality assurance, compliance enforcement, diagnostics, or automated recovery.

In the context of financial underwriting, quality assurance agents can verify extracted customer data and cross-check it against compliance requirements. Diagnostic agents can inspect inconsistencies or missing data, trace the issues to the responsible module, and generate a structured error report that informs corrective actions.

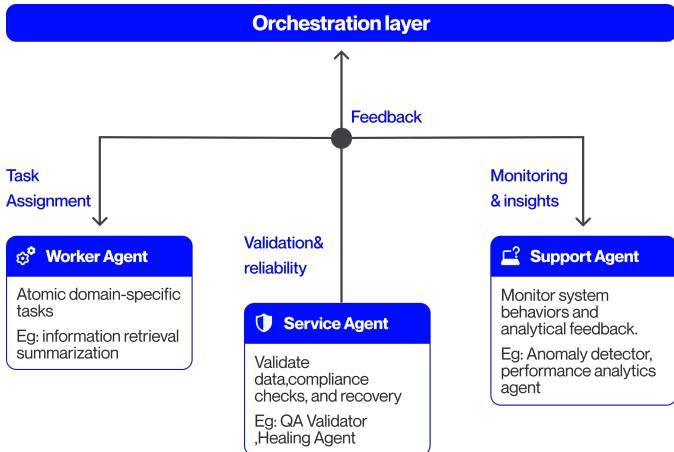


Fig. 2. Specialized agents in a Multi-Agent System

Healing agents can extend this capability by rerunning failed extractions or resetting workflow states to restore normal operation. While healing agents focus on the active system state, upgrade scheduler agents manage version transitions of scoring components and related workflows, ensuring that updates are deployed seamlessly without disrupting ongoing operations.

- **Support Agents** - Complementing the service agents, support agents operate at a supervisory and analytical level. While service agents provide in-line operational utilities during workflow execution, support agents focus on meta-level oversight by monitoring system behavior, analyzing outcomes, and managing data flows that inform orchestration and optimization. Their function is to maintain the overall health, transparency, and adaptability of the system under varying operational conditions. In the financial underwriting scenario, monitoring agents track decision latency, detect risk model drift, and visualize overall portfolio health for both AI orchestrators and human supervisors. Analytics agents evaluate approval-rate patterns and compliance anomalies, while data agents refresh applicant datasets to maintain currency and accuracy.

Fig. 2 summarizes the discussed categories of specialized agents in a multi-agent systems. Across these categories of specialized agents, recent research [9] and enterprise applications demonstrate that coordinated role differentiation significantly enhances the reliability and scalability of multi-agent systems.

## V. ORCHESTRATION LAYER FOR COORDINATED MULTI-AGENT OPERATIONS

The agent orchestration layer forms the control plane of a multi-agent system, transforming autonomous components into a coherent, goal-directed collective. Without orchestration, even highly capable agents risk duplication of effort, logical inconsistency, or unbounded autonomy that diverges from the system's objectives. It interprets system-level objectives,

decomposes them into actionable subtasks, coordinates their execution, and ensures that every output produced aligns with policy, context, and quality requirements.

### A. Planning and Policy Management

At the heart of the orchestration layer are the planning and policy units, which convert high-level objectives into a structured execution plan.

Their purpose is twofold: the planning unit operates as a goal-decomposition engine that determines what tasks need to be done and in what order, while the policy unit embeds domain and governance constraints to define how these tasks are performed. In a financial underwriting workflow, the planning unit assigns tasks such as data extraction and credit scoring to the corresponding worker agents. The policy unit defines operational and governance constraints and coordinates with the control unit (described in the following subsection) to ensure service agents enforce them throughout execution

Together, they translate abstract goals into a directed execution model that defines who performs which task, in what sequence, under what rules, and with what oversight.

### B. Execution and Control Management

Once tasks are planned and assigned, the orchestration layer operates as a distributed control system that transitions specialized agents through phases of initialization, execution, validation, and completion.

Here, the execution unit ensures the smooth operation of the designated tasks performed by worker agents and manages telemetry data collection by support agents. The telemetry data is relayed to the control unit, which may delegate remediation or verification tasks to service agents to maintain operational stability. The control unit also manages concurrency and dependency across workflows, allowing parallel execution and synchronization at key checkpoints to preserve consistency. Additionally, it handles task prioritization and dynamic resource allocation to balance throughput, cost, and determinism across varying workloads.

### C. State and Knowledge Management

For the control unit to achieve synchronization and maintain continuity across workflows, the orchestration layer relies on the state and knowledge management component. This component functions both as a data bus and a knowledge repository.

The state unit manages checkpoints, workflow progress, agent states, and activity logs. Support agents monitor state changes and performance anomalies, while service agents may be invoked to restore checkpoints from the state unit to preserve workflow integrity. The knowledge unit manages contextual and domain-specific information by connecting to external data sources and exposing them as a retrievable context. This ensures that worker agents and orchestration components operate with accurate and aligned information.

This separation of operational state from knowledge state preserves modularity, contextual consistency, and system coherence.

#### D. Quality and Operations Management

Using telemetry, state updates, and contextual data generated by other orchestration units, the quality and operations management component evaluates system performance, validates outcomes, and ensures that orchestrated activities remain compliant and continually optimized. While the control unit focuses on execution stability and the policy unit defines and enforces operational constraints during execution, this component governs verification and optimization of results after execution across the orchestration layer.

The component validates aggregated outputs against defined schemas before integrating them into the shared state, preventing invalid data from propagating through workflows. When inconsistencies or violations are detected, it updates the state accordingly and may invoke service agents to perform diagnostic or remediation actions, ensuring sustained integrity and compliance.

The subsystem also monitors metrics such as latency, throughput, and success rate, using anomaly detection to identify deviations and trigger preemptive interventions. It also supports controlled deployment, testing, and sandboxing new components to maintain stability as agents evolve. Together, these mechanisms ensure resilient, auditable, and continuously improving multi-agent operations.

#### E. Closing Discussion

The orchestration model is exemplified in a financial institution's credit-risk and fraud detection workflow, where specialized agents are coordinated to ensure consistency and compliance. Incoming loan applications are decomposed by the planning unit into subtasks such as data extraction, risk assessment, compliance review, and fraud screening, while the policy unit embeds governance constraints, including lending regulations and institutional risk thresholds. The execution and control component manages concurrent task execution, collects telemetry, and invokes service agents for recovery when needed. The state and knowledge management component maintains applicant states, historical records, and regulatory references for contextual continuity, while the quality and operations component validates results against policy criteria and applies performance insights to optimize future workflows.

Collectively, these mechanisms show that reliability in multi-agent systems arises not only from intelligent agents but from the orchestration layer that governs planning, execution, and validation, enabling scalable and policy-compliant performance.

## VI. COMMUNICATION PROTOCOLS IN ORCHESTRATED SYSTEMS

Building on the orchestration framework, the agent communication layer operationalizes coordination by enabling agents and external tools to exchange information, control signals, and shared context. While orchestration defines who acts and when, communication ensures those actions

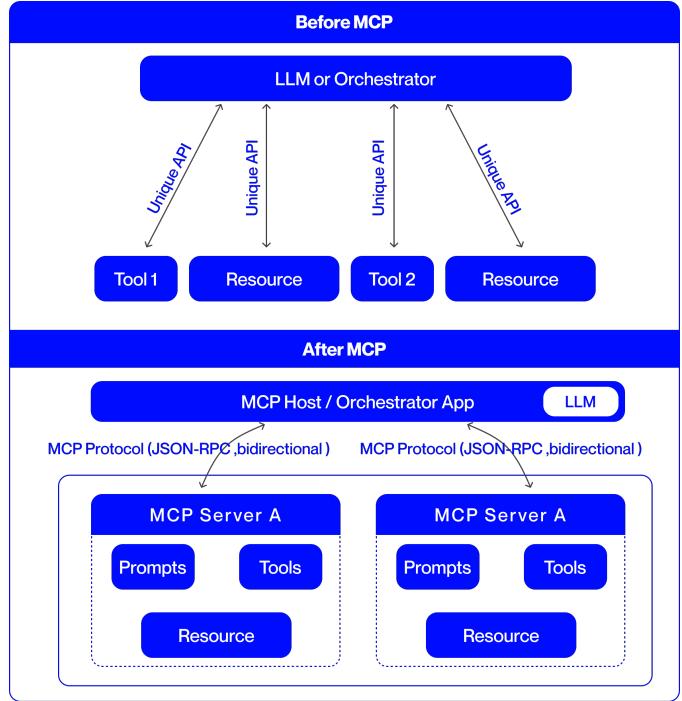


Fig. 3. Comparison of agent communication without MCP and with MCP

remain synchronized and interpretable. Traditional protocols relied on static request-response exchanges and lacked mechanisms for context sharing or policy enforcement. To address these limitations, two emerging standards—the Model Context Protocol (MCP) and Agent-to-Agent (A2A) protocol—establish structured, interoperable communication for tool interaction and peer specialized agentic collaboration. Later subsections explore these protocols in more detail.

#### A. Model Context Protocol

The Model Context Protocol (MCP) [10] provides the standardized communication interface that operationalizes the previously described orchestration flow between agents and external systems like tools, data services, and contextual repositories. As illustrated in Fig. 3, MCP mediates every external invocation through a defined interface that enforces schema consistency, access control, and auditability. This allows the execution unit to dispatch tasks confidently, ensuring that each agent's external interactions conform to orchestration policies and operational rules.

Within the orchestration architecture, MCP follows a client-server design in which agents or orchestrators act as clients that request external capabilities such as tools, resources, or prompts, while connected systems expose these as standardized, callable services. Through MCP, the planning and control units translate defined tasks into executable tool calls and govern how and when agents access these resources in compliance with policy constraints. MCP's session management supports both stateless and stateful exchanges, allowing context continuity across multi-step

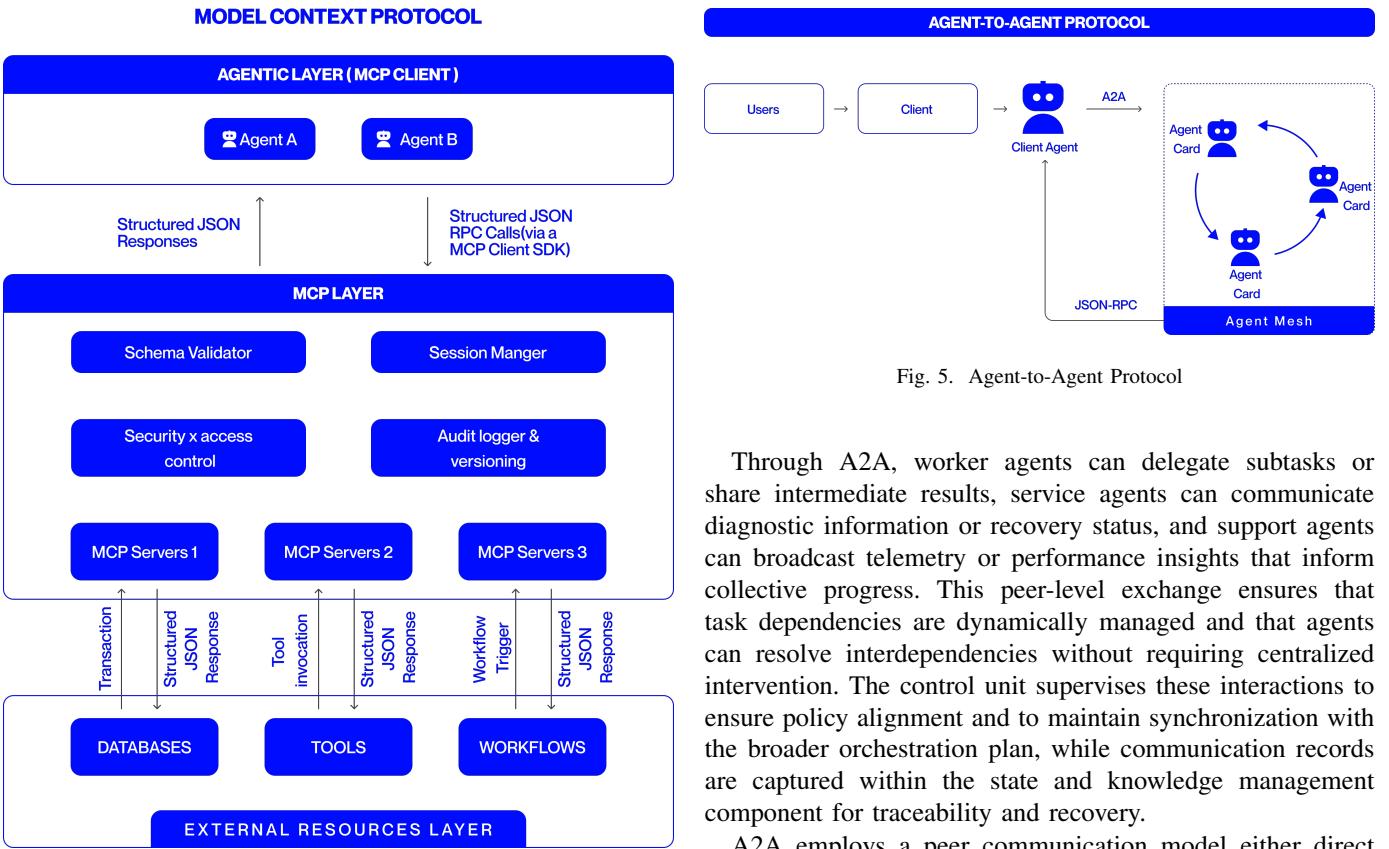


Fig. 4. Integration of the MCP within the orchestration architecture

workflows. These exchanges are logged and synchronized with orchestration state, enabling the state and knowledge management component to maintain consistency and allowing the quality and operations component to verify compliance and alignment with expected results.

As shown in Fig. 4, MCP functions as the operational bridge between high-level orchestration plans and low-level tool execution. It converts planned objectives into structured, policy-aligned invocations and feeds execution data back into orchestration memory and quality loops. Extensions such as ScaleMCP [11] dynamically synchronize tool inventories across agents, while AgentMaster [12] integrates MCP with inter-agent communication frameworks such as A2A to support multimodal collaboration and information retrieval within orchestrated multi-agent systems.

#### B. Agent-to-Agent Protocol

While MCP governs how agents interact with tools and data, the Agent-to-Agent (A2A) protocol [13] defines standardized communication amongst specialized agents themselves. It supports negotiation, delegation, and coordination across distributed ecosystems while maintaining interoperability, traceability, and security [14]. Together, MCP and A2A form the dual foundation of agent communication—MCP for tool access and A2A for peer collaboration (Fig. 5).

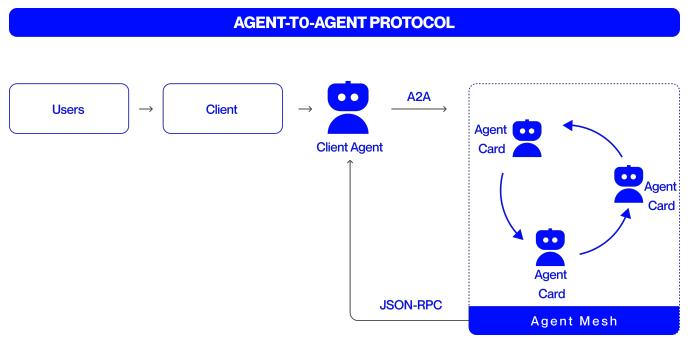


Fig. 5. Agent-to-Agent Protocol

Through A2A, worker agents can delegate subtasks or share intermediate results, service agents can communicate diagnostic information or recovery status, and support agents can broadcast telemetry or performance insights that inform collective progress. This peer-level exchange ensures that task dependencies are dynamically managed and that agents can resolve interdependencies without requiring centralized intervention. The control unit supervises these interactions to ensure policy alignment and to maintain synchronization with the broader orchestration plan, while communication records are captured within the state and knowledge management component for traceability and recovery.

A2A employs a peer communication model either direct or mediated through the orchestrator, enabling reliable, authenticated message exchange. Each message carries structured metadata and standardized payloads, ensuring consistency across heterogeneous implementation. Robust security controls, including cryptographic signing and role-based routing, guarantee message integrity and policy compliance.

Although peer-oriented, A2A remains supervised by the orchestration layer, which validates and synchronizes exchanges to maintain coherence with global workflows. Emerging research explores scalable and hybrid architectures that combine A2A and MCP for multimodal, adaptive coordination in enterprise-grade agentic systems.

## VII. SAFETY, GOVERNANCE AND OBSERVABILITY

Ensuring the reliability of multi-agent systems depends on safeguards embedded within orchestration and communication mechanisms. Within the orchestration layer, the control and quality and operations management units enforce safety and governance through validation, monitoring, and recovery mechanisms that maintain compliance and operational integrity. Similarly, MCP and A2A protocols embed protective measures such as schema validation, authenticated exchanges, and access control to ensure secure and interpretable communication. Core guardrails mitigate hallucinations and enforce consistency checks to prevent agents from producing unsafe or conflicting outputs. These protections are reinforced by internal audits, event logging, and least-privilege policies that promote transparency, accountability, and traceability.

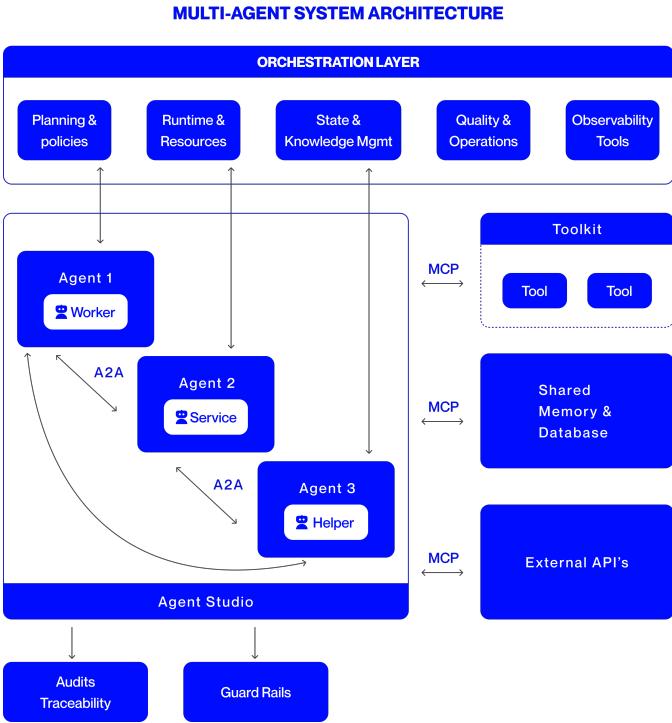


Fig. 6. Orchestrated Multi-Agent System Architecture

across the system. Privacy constraints restrict agents to sharing only task-relevant information. Continuous monitoring, carried out through support agents and the quality and operations management unit, tracks latency, throughput, and correctness to evaluate performance and detect drift. Together, these practices transform multi-agent systems from experimental collectives into dependable, auditable, and continually improving infrastructures that balance autonomy with control.

To summarize, the overall architecture of an orchestrated multi-agent system is shown in Fig. 6. The architecture integrates all core components that enable coordination, communication, and governance across distributed agents. At its foundation are specialized agent types, interacting through standardized protocols such as MCP for tool and data access and A2A for inter-agent collaboration. The orchestration layer oversees planning, execution, and quality control, while observability and governance modules ensure reliability, compliance, and transparency. Together, these elements form a cohesive, scalable framework that operationalizes autonomous intelligence under structured orchestration.

## VIII. CASE STUDIES

### A. Banking, Financial Services and Insurance

Multi-agent AI systems are revolutionizing Banking, Financial Services, and Insurance (BFSI) industry by delivering dramatic efficiency gains and ROI. Insurers are deploying networks of specialized AI agents to automate the labor-intensive underwriting process. For example, autonomous agents studied in [15] now parse insurance applications and supporting documents with over 95%

accuracy, enabling much faster policy issuance. In another use-case explored in [15], a mortgage lender integrated Document AI and Decision AI agents to handle loan paperwork, achieving a 20x faster approval process while cutting processing costs by 80%. Similarly, [16] presents a multi-agent automation framework for property claims underwriting, where specialized agents collaborate to evaluate claim documents, assess damage estimates, and validate policy conditions. Such multi-agent systems clearly outperform both manual processes and single-agent automation, delivering a strong value proposition in BFSI.

### B. Software Engineering and IT Modernization

Multi-agent systems are also proving their worth in software engineering. A large bank recently applied an agentic AI “digital factory” [17] to modernize its legacy core software, which comprised hundreds of applications. Different agents took on specialized coding tasks: one agent automatically documented existing legacy code, another generated new code modules, others reviewed code written by their peers, and additional agents integrated and tested these modules. This multi-agent architecture allows parallel execution and continuous code quality checks, reducing the coordination burdens that slowed the purely human teams. In practice, this approach led to over a 50% reduction in development time and effort for early-adopter teams at the bank.

### C. Cross Industry Adoption

The success of multi-agent AI is spurring widespread adoption across industries. In customer service, companies are reimagining call centers with agentic AI: instead of just assisting human representatives, agents autonomously handle routine inquiries and issues. Studies suggest that up to 80% of common support incidents could be resolved by AI agents without human intervention, cutting resolution times by 60–90% in fully agent-driven workflows. Meanwhile, sectors like healthcare are exploring multi-agent setups where one agent analyzes patient symptoms or medical literature and another suggests treatment plans, all under a doctor’s supervision. From finance and insurance to software development, legal research, and healthcare, multi-agent systems are being rapidly embraced as organizations recognize their measurable performance advantages over manual or single-agent approaches.

## IX. CHALLENGES, RISKS AND FUTURE RESEARCH

As multi-agent systems scale, key challenges emerge around efficiency, cost, and governance. Coordination among numerous agents can create communication overhead, message congestion, and performance bottlenecks unless workflows are carefully managed. The cost of adoption remains significant, enterprises must invest in orchestration software, skilled engineering teams, and continuous monitoring infrastructure to ensure reliability and compliance. Governance presents another difficulty, as decentralized autonomy complicates oversight and accountability. Risks inherited from large

language models, such as hallucination, bias, and data leakage, are magnified when agents interact, raising safety, ethical, and privacy concerns that demand rigorous evaluation and control frameworks.

Future research focuses on making orchestration smarter, safer, and more adaptive. Hybrid and federated designs aim to balance centralized control with decentralized flexibility, while semantic orchestration seeks to match tasks dynamically with the most capable agents. Advances in federated learning and cross-domain collaboration promise secure knowledge sharing without exposing raw data. Standardized benchmarks, simulation testbeds, and open-source orchestration frameworks will further enable transparent performance comparison and lower entry barriers. Together, these directions move multi-agent systems toward scalable, accountable, and trustworthy deployment across enterprise and societal domains.

## X. CONCLUSION

Agentic systems have evolved from single agents that perform narrow tasks, to loosely coupled multi-agent setups, and now to orchestrated collectives where coordination ensures consistency, scale, and reliability. Recent advances show that orchestrated systems are not only viable but already delivering value in real deployments, from BFSI claims processing and fraud detection to healthcare diagnostics and software engineering. Benchmarks and case studies demonstrate measurable gains in productivity, error reduction, and scalability compared with manual or single-agent approaches.

Looking forward, enterprises are moving toward dynamic ecosystems where agents can form, dissolve, and reorganize in response to tasks, much like human teams. To realize this vision, the community must invest in open protocols for interoperability, standardized benchmarks, and shared research infrastructure. With these foundations, orchestrated multi-agent systems can mature into a reliable and adaptable backbone for enterprise intelligence at scale.

## REFERENCES

- [1] Z. Hou, J. Tang, and Y. Wang, “HALO: Hierarchical Autonomous Logic-Oriented Orchestration for Multi-Agent LLM Systems,” *arXiv:2505.13516*, 2025. [Online]. Available: <https://arxiv.org/abs/2505.13516>.
- [2] Y. Dang, C. Qian, X. Luo, J. Fan, Z. Xie, R. Shi, W. Chen, C. Yang, X. Che, Y. Tian, X. Xiong, L. Han, Z. Liu, and M. Sun, “Multi-Agent Collaboration via Evolving Orchestration,” *arXiv:2505.19591*, 2025. [Online]. Available: <https://arxiv.org/abs/2505.19591>.
- [3] PwC, “PwC launches AI Agent Operating System for enterprises,” PwC US Newsroom, Press Release, Mar. 18, 2025. [Online]. Available: <https://www.pwc.com/us/en/about-us/newsroom/press-releases/pwc-launches-ai-agent-operating-system-enterprises.html>. Accessed: Oct. 9, 2025.
- [4] Accenture, “Accenture Introduces Trusted Agent Huddle to Allow Seamless, First-of-its-Kind, Multi-System AI Agent Collaboration Across the Enterprise,” Accenture Newsroom, Jan. 21, 2025. [Online].
- [5] R. Deng, M. Zhao, X. Zheng, H. Guo, and Z. Jin, “Agent Laboratory: Using LLM Agents as Research Assistants,” *arXiv:2501.04227*, 2025. doi: 10.48550/arXiv.2501.04227. [Online]. Available: <https://arxiv.org/abs/2501.04227>.

- [6] S. Luo, Y. Xu, H. Zhang, and K. Zhang, “AgentCoder: Multi-Agent Code Generation with Effective Testing and Self-optimisation,” *arXiv:2312.13010*, 2023. doi: 10.48550/arXiv.2312.13010. [Online]. Available: <https://arxiv.org/abs/2312.13010>.
- [7] Y. Lin, J. Park, X. Li, and S. Zhang, “Multi-Agent Fact Checking,” *arXiv:2503.02116*, 2025. doi: 10.48550/arXiv.2503.02116. [Online]. Available: <https://arxiv.org/abs/2503.02116>.
- [8] Z. Zhao, Y. Sun, H. Wu, L. Zhang, and Y. Li, “KoMA: Knowledge-driven Multi-agent Framework for Autonomous Driving with Large Language Models,” *arXiv:2407.14239*, 2024. doi: 10.48550/arXiv.2407.14239. [Online]. Available: <https://arxiv.org/abs/2407.14239>.
- [9] Y. Fu, H. Sun, Y. Song, J. Liu, X. Chen, Z. Zhang, J. Tang, and T.-S. Chua, “A Survey on Agentic Large Language Models: Architecture, Capabilities, and Applications,” *arXiv preprint arXiv:2402.01680*, 2024. [Online]. Available: <https://arxiv.org/abs/2402.01680>. Accessed: Nov. 17, 2025.
- [10] Model Context Protocol, “What is the Model Context Protocol (MCP)?” Documentation, 2025. [Online]. Available: <https://modelcontextprotocol.io/docs/getting-started/intro>. Accessed: Oct. 9, 2025.
- [11] E. Lumer, A. Gulati, V. K. Subbiah, P. H. Basavaraju, and J. A. Burke, “ScaleMCP: Dynamic and Auto-Synchronizing Model Context Protocol Tools for LLM Agents,” *arXiv:2505.06416*, 2025. doi: 10.48550/arXiv.2505.06416. [Online]. Available: <https://arxiv.org/abs/2505.06416>.
- [12] C. C. Liao, D. Liao, and S. S. Gadiraju, “AgentMaster: A Multi-Agent Conversational Framework Using A2A and MCP Protocols for Multimodal Information Retrieval and Analysis,” *arXiv:2507.21105*, 2025. doi: 10.48550/arXiv.2507.21105. [Online]. Available: <https://arxiv.org/abs/2507.21105>.
- [13] A. Chawla, “A Visual Guide to Agent2Agent (A2A) Protocol,” *Daily Dose of Data Science* (Substack), Apr. 16, 2025. [Online]. Available: <https://blog.dailydoseofds.com/p/a-visual-guide-to-agent2agent-a2a>. Accessed: Oct. 9, 2025.
- [14] The Linux Foundation (A2A Project), “Agent2Agent (A2A) Protocol — Latest Documentation,” 2025. [Online]. Available: <https://a2a-protocol.org/latest/>. Accessed: Oct. 9, 2025.
- [15] B. Pazur, “17 Useful AI Agent Case Studies,” *Multimodal.dev* (blog), May 14, 2025. [Online]. Available: <https://www.multimodal.dev/post/useful-ai-agent-case-studies>. Accessed: Oct. 9, 2025.
- [16] M. I. Sajid, “Multi-Agentic Automation for Evaluating Property Claims in Underwriting,” *Open Journal of Applied Sciences*, vol. 15, no. 4, pp. 819–833, Apr. 2025. doi: 10.4236/ojapps.2025.154055. [Online]. Available: <https://www.scirp.org/journal/paperinformation?paperid=141685>.
- [17] A. Sukharevsky, D. Kerr, K. Hjartar, L. Hämäläinen, S. Bout, and V. Di Leo, “Seizing the agentic AI advantage,” McKinsey & Company (QuantumBlack) report, Jun. 13, 2025. [Online]. Available: <https://www.mckinsey.com/capabilities/quantumblack/our-insights/seizing-the-agentic-ai-advantage>. Accessed: Oct. 9, 2025.