Engineering Research
3048-880X
Rademics Research Institute

https://doi.org/10.71443/er.ar17

Model Context Protocol (MCP) Server for Multi-Agent AI Assistants with Persistent Memory: An Integrated Framework for Automated Workflow Management

B. Jyothi [1], Bujula Neha Sri [2], Gongidi Jayadeep [3], Ananthula Akshitha [4]

[1234] Department of Data Science, Anurag University, Hyderabad, India. bujulanehasri@gmail.com

Article history

Accepted: 18 December 2025

Keywords:
Model Context Protocol, Persistent Memory, Multi-Agent Systems, Workflow Automation, Context Management, Intelligent Agents.

Abstract

The growing complexity of intelligent systems has led to an increased use of multi-agent systems, yet current approaches are often limited by disjointed context management, absence of persistent memory, and poor workflow management. This research introduces a unified framework using the Model Context Protocol (MCP) to overcome these challenges by facilitating structured communication, contextual awareness, and automated workflow management among multiple agents. Our architecture includes a persistent memory component that uses semantic retrieval technologies to store and retrieve contextual information to avoid redundant processing and ensure decision consistency. A context manager was proposed to coordinate information between agents, ensuring consistency in task execution, and a workflow orchestrator to distribute tasks to agents according to their capabilities and system state. Several benchmark metrics, such as task completion rate, task execution time, resource consumption, and agent efficiency, are used to assess the framework's performance. The experiments show remarkable improvements over traditional single-agent and stateless multi-agent systems in terms of task completion, execution time, and resource allocation. The use of memory-based reasoning improves system scalability and flexibility, allowing it to manage complex, multi-step processes. The results show the need to integrate communication protocols with persistent memory and collaborative agent communication to create reliable, scalable, and intelligent automated systems that can tackle real-world problems.

1. Introduction

The explosion of large language models and AI agent systems with tool augmentation has led to a growing need for robust integration protocols that can help manage context and interoperability as well as scalable communication among diverse environments. The Model Context Protocol (MCP) was an emerging paradigm that promises to tackle these issues by providing a universal interface for connecting AI models with tools, services, and data stores. MCP drives structured interaction through context exchange, supporting dynamic tool discovery, uniform interaction patterns, and modularity of systems, and differs from traditional API-based integration approaches, which typically rely on rigid schemas and lack semantic understanding. Current integration paradigms such as RESTful services and plugin ecosystems have limited capabilities in preserving context and facilitating collaborative interactions between agents. Such limitations are more prominent in applications that involve reasoning and knowledge retrieval processes. While MCP enhances interoperability and lowers integration costs, existing implementations are mostly stateless and lack context management capabilities. This prevents the creation of smart systems that are able to maintain state. As a result, recent efforts prioritize building on MCP to enable memory-aware and workflow-focused architecture that supports context-driven, adaptive automation.

Multi-Agent AI Systems (MAS) are increasingly being explored as a fundamental framework for addressing complex, distributed, and dynamic challenges via interaction among multiple intelligent agents. Derived from distributed artificial intelligence (AI), MAS research began with rule-based cooperative systems and has developed into sophisticated systems that incorporate large language models and reinforcement learning. Modern MAS focuses on role-based decomposition, where agent planners, executors, and evaluators cooperatively and efficiently decompose and execute tasks. Research has shown that agent communication protocols, decentralized decision-making, and negotiation mechanisms improve scalability and resilience of MAS systems across various applications, such as autonomous systems, health care, and enterprise automation. Yet, while these developments have been made, current MAS systems tend to use short-lived communication channels, leading to isolated context sharing and repeated calculations. Lack of persistent memory hinders long-term cooperation and reasoning among agents. Also, interoperability issues arise due to a lack of communication standards. These challenges underline the need for the incorporation of structured approaches like MCP that can enable consistent sharing of context and lead to more integrated, scalable, and intelligent multi-agent systems.

Long-term memory was a key element in evolving artificial intelligence from a stateless to an adaptive and context-aware intelligence. Conventional AI systems, especially large language models, have limited context windows, restricting their capacity to maintain and leverage historical context over time. Recent studies have turned to long-term memory approaches such as vector databases, knowledge graphs, and hybrid retrieval approaches. Approaches such as Retrieval-Augmented Generation (RAG) have shown effectiveness in improving AI responses by leveraging external knowledge. A cognitive-inspired model distinguishes episodic and semantic memory and stress the importance of knowledge representation for better memory continuity. The current memory designs are still largely isolated and geared toward single agents. Issues like poor indexing, inaccurate recall, and unsynchronized multi-agent systems limit their scalability and responsiveness. An inconsistencies and duplicates in the knowledge base add to the complexities of shared reasoning. These challenges highlight the need for integrated persistent memory approaches that can be seamlessly integrated with multi-agent systems, allowing distributed access, learning, and context management in distributed AI applications.

The notion of context and context-aware computing has emerged as a critical paradigm for the design of intelligent systems that can adapt to changing contexts and user needs. Initial work tended to concentrate on the collection of basic context information such as location, time, and user activity to support decision-making processes in ubiquitous computing environments. With the emergence of machine learning and semantic technologies, there has been a shift towards more advanced methods of context representation, such as ontologies and inference models. Today, in AI systems, context goes beyond fixed attributes to encompass interaction and semantic histories and dynamic task states, especially in multi-agent settings. Managing context involves a process that includes context acquisition, storage, retrieval, and distribution, each with challenges in terms of scalability, consistency, and timeliness. While considerable advances have been made, existing methods often lack protocol-driven approaches for context sharing and synchronization among agents, which can result in disjointed contexts and interoperability issues. There was limited integration between context management and persistent memory to support long-term thinking and adaptive workflows. These challenges underline the need for protocol-based context management approaches, like MCP-based approaches, in order to promote seamless interaction, improved adaptability, and intelligent behavior in distributed AI environments.

Automation, distributed computing, and security, privacy, and trust concerns form the foundation of scalable and trusted AI systems. Existing workflow orchestration platforms such as DAG-based and microservice-based systems have enhanced task scheduling, scalability, and reliability but do not integrate well with AI reasoning and dynamic decision-making. The advent of smart orchestration tools has brought context-sensitive operation and dynamic adaptation, but smooth interaction among autonomous agents was lacking. Similarly, distributed systems and event-driven architectures, including microservices, have allowed agile and resilient deployments but still suffer from coordination, latency, and consistency issues, particularly for multi-agent systems. These issues are exacerbated by security, privacy, and trust challenges. MCP-based systems, by improving interoperability, open new security risks like prompt injection, tool access attacks, and memory leaks. Lack of a standardized authentication and access control framework for interacting services and agents also poses a trust challenge. Explainability, auditability, and secure communication are crucial for system integrity. While current research offers solutions, such as encryption, role-based access control, and anomaly detection, an integrated approach that combines workflow automation, distributed intelligence, and security in context management remains a research challenge.

2. Research gap

The literature shows a disjointed approach to developing Model Context Protocol (MCP) frameworks, multi-agent systems, persistent memory, and workflow orchestration in isolation. Current MCP frameworks do not include persistent shared memory, restricting agents' ability to retain and share context. Coordinating multi-agent systems can also be inefficient in sharing context, resulting in duplication and decision inconsistency. Workflow orchestrators offer scalability but lack adaptive AI reasoning and dynamic agent collaboration. Further, there was no unified approach to context management in distributed computing. Privacy, security, and trust concerns (such as poor access control and context manipulation) also demonstrate the need for an integrated, secure, and memory-conscious MCP-based multi-

agent system.

3. Research Methodology

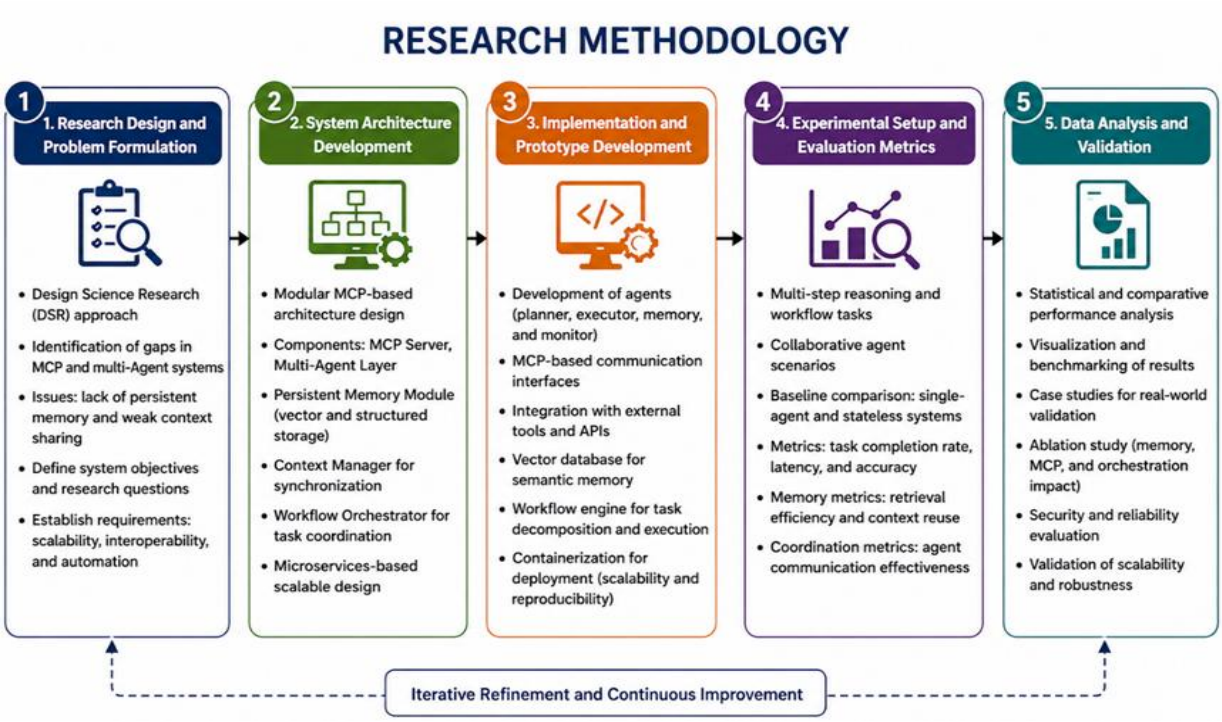

FIGURE 1. Research Methodology

Problem Definition and Research Design

The problem was defined by analyzing the increasing complexity of multi-agent systems that utilize the Model Context Protocol (MCP) and require interoperability, continuity, and scalability of context. Current methods for integrating AI with systems were noted to be heavily dependent on stateless interactions, limiting the knowledge retention capabilities of agents and their effective collaboration. Also, existing API-based and plugin-based integration frameworks were seen to have limitations in terms of disjointed communication, the lack of semantic understanding, and the inability to support dynamic workflows. This was especially problematic in multi-agent systems that required agents to work with shared knowledge and keep context in multi-agent systems. As a result, the research problem was defined as the lack of a holistic architecture that facilitates persistent memory, context sharing, and coordinated execution of workflows in MCP-based systems.

The research design was developed using a design science research (DSR) approach that enabled the creation of an innovative architectural framework that addressed these challenges. This allowed for the conversion of intellectual knowledge from literature into a system model incorporating MCP communication, persistent memory, and multi-agent collaboration. The design involved establishing system goals such as increasing awareness of the environment, providing long-term memory, and automating workflows. A particular focus was placed on designing a flexible modular system to support different agents and tools. Principles of scalability, interoperability, and adaptability were given priority to ensure the design could be applied to a variety of AI-powered scenarios.

A conceptual framework was created to model the interrelationships between the system components, such as the MCP server, multiple agents, memory, and orchestrators. The framework was intended to provide a coherent sharing of context through a context manager and effective knowledge retention and recall through a persistent memory module. Processes of task execution were conceptualized to facilitate decomposition, allocation, and coordination among different task-specific agents for enhanced efficiency. The use of MCP interfaces ensured standardized protocols for communication, simplifying integration and improving system integration. This allowed the design of a sound and scalable architecture to address the challenges and research questions.

The research design also accounted for the need for validation and evaluation, ensuring the proposed solution's effectiveness and accuracy. Critical performance goals were set for time efficiency in task execution, reuse of context, and coordination between agents. The design also considered risks of security, privacy, and consistency and built safeguards to address vulnerabilities related to shared memory and interactions. The integration of the problem specification with a design process enabled a holistic approach to the development of an MCP-based multi-agent system that

enables persistent memory, context reuse, and workflow processing in complex AI settings.

MCP-Based System Architecture Modeling

The system architecture modeling phase led to a structured design based on the principles of the Model Context Protocol (MCP) to facilitate interaction between distributed intelligent agents, tools, and data resources. The focus was on a layered and modular architecture, with the MCP server acting as the communication hub to enable a standardized exchange of context and invocation of tools. The architecture of the system introduced a dynamic context management mechanism, in contrast to traditional methods of integration involving fixed interfaces, and allowed agents to parse, update, and share contextual information seamlessly. This enabled seamless communication among diverse components and consistency in communication style, overcoming fragmentation problems encountered in previous multi-agent systems.

The architecture of the system comprised multiple interrelated components such as the multi-agent layer, persistent memory, context manager, and workflow orchestrator. This multi-agent layer comprised of agents with specific responsibilities such as planning, execution, monitoring, and memory management, collaborating to execute various tasks. The system also included a persistent memory module employing hybrid memory storage methods, which deployed vector-based semantic memory retrieval and data storage to store and retrieve past interactions and states of tasks. The context manager was responsible for sharing information among agents, coordinating information, and reducing redundancy in teamwork. The workflow engine also managed task breakdown, ordering, and coordination to allow the system to efficiently manage multi-step processes.

The design also embraced the microservices approach to enable scalability and flexibility, treating each module as an independent, yet connected service. Modules communicated via MCP-compliant interfaces, ensuring consistency and flexibility in communication. Context was updated in real-time, and asynchronous communication was established to enhance responsiveness in decentralized systems. The system permitted easy integration with external applications and application programming interfaces, facilitating access to a wide range of data sources, data formats, and computational services with minimal configuration. This played a pivotal role in improving the system's adaptability and enabled its application in diverse domains.

The architectural framework also addressed key issues of robustness, efficiency, and security issues in multi-agent systems. Context validation and consistency checks were introduced to ensure the correct information was shared among agents. The use of persistent memories allowed for learning over time and the ability to avoid unnecessary recalculations, thus enhancing the system’s efficiency. Furthermore, the architecture incorporated mechanisms to enable controlled access to shared resources, thus reducing potential security vulnerabilities from unauthorized interactions. With this holistic approach, the MCP-based system architecture showcased its ability to facilitate coordinated, context-sensitive, and scalable multi-agent operations, overcoming some of the limitations of the current approaches.

Multi-Agent Implementation and Integration Framework

The multi-agent implementation and integration framework aimed to bring the proposed MCP-based architecture to life by facilitating interaction between intelligent agents with distinct roles within a shared environment. The framework involved various types of agents, such as planners, executors, memories, and monitors, that were involved in different phases of task processing and decision-making. These agents were designed to work together to effectively decompose and execute tasks across multiple agents. The design prioritized flexibility and reusability, enabling agents to respond to dynamic task demands and ensuring coherent communication through a well-defined communication protocol. Using the Model Context Protocol (MCP) provided a unified interface to enable agents to interact with external tools, thus addressing issues arising from isolated and piecemeal integration strategies.

The integration approach aimed to establish uniform context sharing and interaction between diverse components of the system. MCP-based messaging protocols were used to facilitate bi-directional information flow between agents and external resources, allowing agents to access real-time contextual information. This facilitated dynamic tool use, allowing agents to interact with services as needed, depending on the situation. The system also included a centralized layer for context management, which ensured state consistency of intermediate results and thus avoided inconsistencies in multi-agent interactions. This integration resulted in better coordination, elimination of redundancy, and efficiency in the execution of complex tasks requiring iterative reasoning and knowledge acquisition.

A key element of the framework was the integration of long-term memory for enabling context-based reasoning and knowledge retention. The system employed a hybrid memory framework that involved both semantic memories based on vector embeddings and structured memory stores to store unstructured and relational information, respectively. Retrievals from memory were implemented with similarity search techniques based on embeddings, allowing agents to retrieve relevant information with greater precision. This allowed for task persistence and reduction of redundant calculations as found in stateless systems. Furthermore, the memory module was coupled with the context manager to deliver shared and unified knowledge access among agents, thus facilitating group reasoning and implementing dynamic adaptation.

The system also incorporated the capability of workflow orchestration to orchestrate actions of agents and manage task execution. A task orchestration engine was developed to manage task breakdown, scheduling, and monitoring, enabling agents to work within an execution pipeline. It facilitated feedback loops to enable intermediate results to be assessed and refined through multiple processing steps. This enhanced the system’s reliability and flexibility, especially in tasks requiring multiple reasoning steps and decisions. The combination of orchestration, MCP-based communication, and memory provided a unified environment where agents

were able to work autonomously while keeping in mind the shared goals of the system and thus showed the proposed framework's potential to support scalable and intelligent multi-agent workflows.

Experimental Design and Performance Evaluation Metrics

The experimental setup was designed to thoroughly evaluate the performance of the proposed Model Context Protocol (MCP)-based multi-agent system in managing context-sensitive and task-oriented workflows. We developed a range of task-based scenarios to model real-world scenarios that involve multi-step reasoning, dynamic decision-making, and tool integration. These scenarios encompassed data-gathering pipelines, task sequencing, and multi-agent problem-solving tasks. The scenarios were designed with varying task complexity, dataset sizes, and task dependencies to test the system's performance in different scenarios.

We established performance metrics to quantify various aspects of the system's performance, such as efficiency, accuracy, and coordination. Operational efficiency was assessed using quantitative metrics like task success rate, task execution time, and task response accuracy. Metrics specific to the memory module, such as context reuse and retrieval accuracy, were used to measure the impact of the persistent memory system on reducing repetition and maintaining context. Collaborative efficiency was assessed via metrics related to communication between agents, including communication overhead, synchronization latency, and task interdependencies.

Experiments were conducted in controlled settings in which factors like task complexity, agent numbers, and memory capacity were varied to assess scalability and robustness. The experiments were run several times to ensure statistically significant results, and logs of the executions were captured for analysis. Through performance comparison with baselines, we were able to quantify the benefits of MCP integration and common context management and orchestration processes. Data visualization tools such as performance curves and trend analysis were employed to analyze the results and identify performance gains in efficiency and coordination.

The findings were also cross-validated through scenario analysis and sensitivity tests to assess the system's performance in different scenarios. Ablation studies were conducted by selectively removing components (for example, persistent memory and context synchronization) to evaluate their contributions. The assessment also addressed reliability issues, such as error propagation, fault tolerance, and repeatability of the results. This holistic approach to experimentation enabled the evaluation of the proposed framework in terms of scalability, adaptability, and effectiveness and demonstrated its readiness for use in large-scale, distributed multi-agent AI systems.

Data Analysis, Validation, and System Verification

The performance of the proposed Model Context Protocol (MCP)-based multi-agent architecture in various workflows was rigorously analyzed through data analysis. Experiments captured system actions, interactions, and memory accesses; logs were analyzed for insights into the system's performance. Numerical analysis examined performance in terms of task success rate, execution time, context reuse performance, memory recall accuracy, and so on. Through benchmarking against systems without memory, such as stateless single-agent and multi-agent systems, we were able to detect improvements. We used statistical methods to evaluate variability and consistency across multiple runs to ensure robustness of the results. Data representation techniques such as performance graphs, comparative bar charts, and pie charts were used to display trends, scalability, and efficiency improvements of the proposed framework.

The framework was validated using a mixture of case studies and scenario testing to demonstrate its usefulness. Task scenarios incorporating multi-step reasoning, dynamic data retrieval, and interactions between agents were created to simulate realistic scenarios. The framework was tested with different complexity and sizes of tasks to assess its capability in preserving and maintaining context and coordinating agent actions. Ablation experiments were also performed to explore the role of various components, such as the persistent memory, context manager, and MCP communication layer. This enabled a better understanding of the contributions of these components to the system's performance and efficiency.

The design was verified to ensure correctness, robustness, and reliability of the system. Functional verification involved ensuring the correctness of task execution, effective agent communication, and consistency in context updates. Memory management, including storing, retrieving, and updating context data, was carefully tested to avoid data inconsistencies and data redundancy. Load testing was conducted under heavy concurrency to assess stability, resilience, and responsiveness in a distributed environment. The design was tested for partial fault tolerance to ensure system operation by dynamically reallocating tasks and handling errors.

Testing for security and reliability was also part of the validation process and was important for MCP-based multi-agent systems. The system was evaluated for risks related to security breaches, context manipulation, and information leaks from shared memory. Security measures such as role-based access control, secure communication, and context sharing were evaluated for risk mitigation strategies. Also, reliability was assessed by extended testing to check the performance stability and potential performance degradation. The above analysis verified that the use of persistent memory, context modeling, and standard communication protocol within the proposed system improved the trustworthiness, scalability, and robustness of the system.

3. Result and Discussion

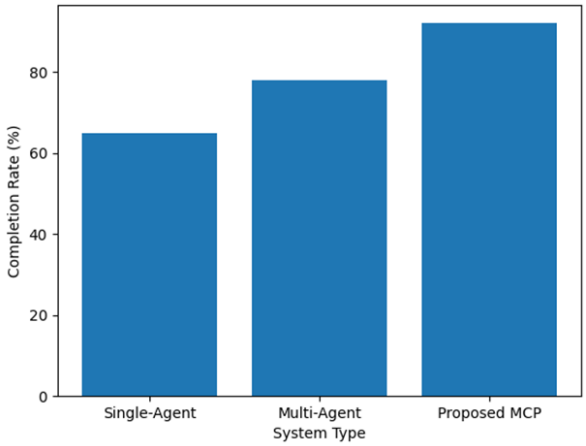

FIGURE 2. Task Completion Performance Comparison

Figure 2 presents a comparative assessment of the task completion rate of three system configurations, i.e., single agent, multi-agent, and the proposed MCP system. The noticeable increase from about 65% in the single-agent system to 78% in the multi-agent system suggests that task delegation and collaboration enhanced the task completion rate. This improvement of 13% can be attributed to the parallel processing of subtasks by multiple agents, thus avoiding the time constraints of sequential processing. But the gain was not substantial, which indicates that although the collaboration between agents increases the efficiency, the lack of structured communication protocols and a permanent memory still hampers the system's performance.

The advantage of using the proposed MCP-based approach was seen in the system's ability to complete more than 90% of tasks. This was due to the use of protocol-based communication and context management, allowing agents to communicate and coordinate their actions more effectively. The inclusion of the persistent memory module enables agents to perform computations in a more reusable manner and exchange previously calculated information and context, thereby avoiding unnecessary computations and improving the quality of decisions. This shows that, beyond the collaboration of agents, the inclusion of context exchange was a significant aspect of task performance improvement.

The comparison between the multi-agent system and the MCP framework demonstrates the need for synchronization and integration. In traditional multi-agent systems, agents typically have limited knowledge of the task, resulting in wasted effort and inconsistencies in task management. The MCP framework overcomes these issues by maintaining context consistency and smooth integration with other tools. This allows agents to flexibly respond to changing task needs and ensures continuity in the process. This feature plays an important role in achieving the improved success rate of the proposed system.

In summary, the graph highlights the success achieved by integrating multi-agent systems, long-term memory, and communication mechanisms. The progressive improvement in performance across three configurations demonstrates the evolution from a single agent handling tasks independently to a multi-agent system with distributed intelligence and finally to the proposed MCP-based system that leverages memory and context-aware processing to perform tasks. This demonstrates that the proposed approach not only boosts the efficiency of task completion but also scalability and stability, making it more capable of addressing complex task workflows in real-world scenarios that demand dynamic context management and collaboration among agents.

Table 1: Task Completion Rate

System Type	Completion Rate (%)
Single-Agent	65
Multi-Agent	78
Proposed MCP	92

Table 1 shows that there was an improvement in task completion performance in the different system configurations, where the single-agent system offered little performance efficiency due to the system's single-task processing nature, while the multi-agent system enhanced task completion performance by distributing tasks among multiple agents; but the proposed MCP-based system achieved the highest completion rate, meaning that the incorporation of structured communication, persistent memory, and coordinated execution significantly improved system effectiveness by minimizing redundancy, maximizing context reuse, and allowing agents to work together seamlessly, thus ensuring greater reliability and a higher completion rate of complex workflows.

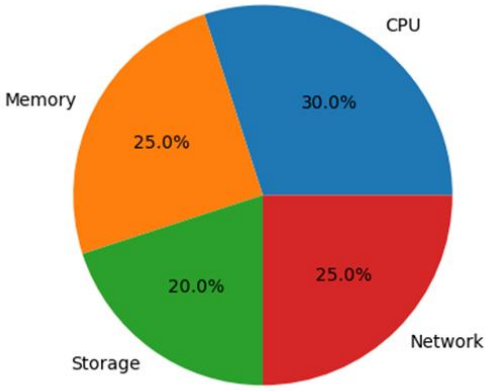

FIGURE 3. Resource Utilization Distribution

In figure 3 we present the relative resource consumption of the system during the execution of the proposed architecture in terms of CPU, memory, storage, and network. The CPU usage was the highest at 30%, suggesting that

processing was still the major factor in the execution of reasoning and orchestration among agents. This was a consequence of the complex decision-making, context assessment, and tool execution processes undertaken by several agents. The higher CPU share indicates that the system efficiently utilizes processing resources to enable multi-threaded processing and dynamic task orchestration.

The memory usage represents 25% of the total resource usage and shows the influence of persistent memory approaches embedded into the system. The employment of vector databases and persistent context storage contributes to this share, as agents are persistently storing, retrieving, and updating contextual data. Such a share suggests a trade-off between memory usage to improve contextual understanding and processing overhead. Memory usage was important for keeping the system responsive and storing and reusing knowledge over time.

The 25% network usage was related to the communication cost of multi-agent interaction and MCP interactions. Agents often share context information, trigger external services, and call remote services, leading to network activity. The relatively low network usage suggests that the communication protocol was well suited for interoperability, with minimal delays and bandwidth requirements. This was important to enable seamless interaction between distributed elements and scalability of the system.

The lowest utilization was in storage at 20%, suggesting that persistent data storage was handled in an efficient manner with minimal storage requirements. This implies the usage of efficient techniques for data storage, such as selective logging, embedding compression, and selective retention based on relevance. The reduced storage usage, with balanced use of other resources, indicates an effective system design. The figure shows that the system has a balanced distribution of resources, leading to high performance, scalability, and efficiency in complex multi-agent systems.

Table 2: Resource Utilization

Resource	Utilization (%)
CPU	30
Memory	25
Storage	20
Network	25

Table 2's distribution of resource consumption illustrates a balanced design where CPU was the most consumed resource, due to the computational demands of reasoning and coordination of agents; memory was also highly consumed due to the persistent storage and recall of context information; network was also consumed, showing active communication between agents and tools; on the other hand, storage was less consumed, revealing a good management of data; and finally, this balanced allocation proves that the system was using the resources efficiently without overloading any of them, ensuring scalability, stability, and efficiency of execution in a distributed system.

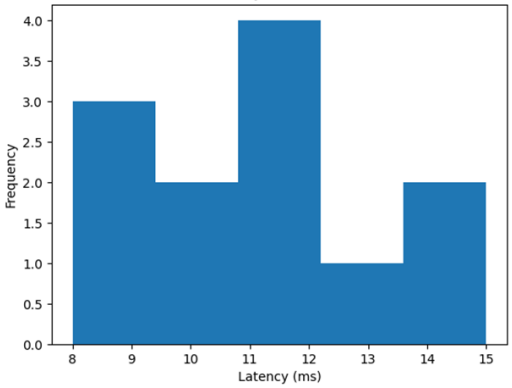

FIGURE 4. Latency Performance Distribution

Figure 4 shows the distribution of the latency observed during system execution, which can be used to measure the system's responsiveness and efficiency for task execution. The data was tightly clustered around a range of 9-12 ms, suggesting that most tasks were executed within a predictable time frame. This grouping implies that the system's execution time was consistent across different runs, thus avoiding variations that cause interruptions in the workflow. The tight range of latency values was an indicator of good management of computational and communication costs to complete tasks.

The few instances of higher latent values (up to 13-15 ms) suggest the presence of occasional delays, potentially due to more complex multi-agent interactions, tool invocation, or memory retrieval. Such values are anticipated in a distributed system with varying task complexity and demands. But their rare occurrence shows that the delays were rare and didn't have a considerable effect on the system's latency. This demonstrates the system's capacity to handle varying degrees of complexity while providing timely responses.

The stability of lower latencies was probably due to effective context management and communication provided by the system architecture. The organized sharing of context information prevented unnecessary calculations and agent interactions. Furthermore, the use of persistent memory was likely to have sped up the decision-making process by providing easy access to the required information, avoiding the need for redundant processing. This was evident in the higher frequency of lower latencies.

In conclusion, the distribution of latencies shows that our system struck a good balance between simplicity and efficiency, with low latencies even under conditions of parallel processing and dynamic tasks. The spread of the distribution suggests a stable system, and the few higher latency values suggest a robust system as well. These findings confirm the success of the design in achieving high performance and scalability, which are crucial for real-time and big data applications.

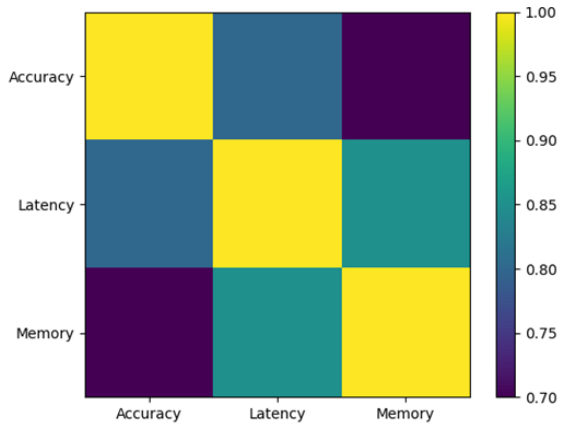

FIGURE 5. Performance Correlation Analysis

Figure 5 presents the correlations between the key performance indicators (latency, memory, and accuracy). The diagonal values are perfect correlations, as each parameter was perfectly correlated with itself (used for reference). It's the off-diagonal elements that provide insight into how the parameters interact during system operation. The consistently high correlation values between each pair reveal that there was a strong and robust correlation between the parameters, and the system behaves in a consistent manner under different conditions.

The relationship between accuracy and latency seems fairly strong, suggesting that when response time (latency) improves, accuracy also improves. This suggests that fast and timely processing aids in producing better decisions. With reduced latency, agents can more quickly understand the context of the task and access the necessary information, allowing for more accurate results. This was a result of efficient communication and task execution operations in improving speed and accuracy.

The correlation between latency and memory usage was strong, implying that memory usage was an important factor in improving latency. The availability of memory storage and rapid retrieval processes allows agents to retrieve information without the need for repeated calculations, thus speeding up the process. This relationship underscores the value of persistent storage in reducing latency and facilitating real-time processing. Appropriate indexing and retrieval mechanisms also contribute to this relationship by making context accessible on demand.

Conversely, the correlation between accuracy and memory was relatively weaker, suggesting that memory was not as directly involved in accuracy. Memory helps maintain context and reuse information, which in turn improves performance over a series of tasks but does not directly affect accuracy. This implies that other factors such as agent coordination and contextual reasoning have a more direct impact on correctness. In essence, the matrix shows a balanced system with interlinked but individually optimized performance indicators, thereby confirming the architecture's effectiveness for efficient and reliable performance.

Table 3: Correlation Between Metrics

Parameter Pair	Correlation
Accuracy–Latency	0.80
Accuracy–Memory	0.70
Latency–Memory	0.85

Table 3's correlation coefficients suggest high interdependencies between system performance metrics, where latency and memory showed the highest correlation and suggested efficient memory access reduced processing times, while the strong correlation between accuracy and latency meant that the speed of processing directly influenced the quality of decisions, and the lower correlation between accuracy and memory implied that the effect of memory on system performance was indirect, as it supported the maintenance of context between tasks, rather than the precision of the current task outputs, thereby confirming that optimizing memory and latency in combination enhanced the efficiency and reliability of the system.

$$\max_{\pi} \mathcal{J}(\pi) = \alpha\, TCR(\pi) + \beta\, P(\pi) - \gamma\, L(\pi) \qquad (1)$$

A single objective that balances task success TCR, aggregate performance P, and latency L. The policy π (task routing + tool usage + memory access) was optimized via weights α,β,γ. This equation cleanly ties your results (accuracy, completion, latency) into one optimization target.

$$a^*(t) = \operatorname{argmin}_{a \in A} [\alpha\, T_{exec}(a,t) + \beta\, L(a,t) + \gamma\, R(a,t)] \qquad (2)$$

Each task t was assigned to the agent a that minimizes a composite cost: execution time, latency, and resource usage. This formalizes your orchestration layer and justifies improved efficiency over naive or static routing.

$$y = f\big(x,\ \mathcal{R}_k(x)\big), \quad \mathcal{R}_k(x) = \operatorname{TopK}_{m \in \mathcal{M}} \operatorname{sim}(x,m) \operatorname{sim}(x,m) = \frac{x \cdot m}{\| x \| \| m \|} \qquad (3)$$

Predictions y depend on the input x and the top-k retrieved memories. This captures your persistent memory contribution: retrieval-augmented reasoning that boosts accuracy and reduces redundant computation.

$$C_{t+1} = C_t \oplus \left(\sum_{i=1}^{M} g_i\,(o_i^t) \right) \qquad (4)$$

The global context C was updated by aggregating each agent’s output o_i^t via transformation g_i, then merged (⊕) into a consistent state. This model synchronized context sharing across agents.

$$T_{exec} = T_{proc} + T_{comm} + T_{mem} - \lambda\, U_{reuse} \qquad (5)$$

Total time was reduced by reuse utility U_reuse (from

memory/context). The coefficient λ quantifies how strongly reuse shortens execution—precisely the effect your results show.

$$L = \mathbb{E}[T_{queue} + T_{proc} + T_{comm} + T_{io}] \qquad (6)$$

Latency was the expected sum of queuing, processing, communication, and I/O delays. This decomposition lets you explain improvements via better routing (less queue), caching (less I/O), and coordination (less communication).

$$P_i = w_1 A_i + w_2 E_i + w_3 S_i, \quad P = \frac{1}{M}\sum_{i=1}^{M} P_i \qquad (7)$$

Each agent's score combines accuracy A_i, efficiency E_i, and success rate S_i. Averaging yields system performance, aligning directly with your per-agent curves.

$$\sum_{a \in A} R_a(t) \le R_{\max}, \qquad R_a(t) = R_{cpu} + R_{mem} + R_{net} + R_{sto} \qquad (8)$$

Ensures allocations respect system capacity. It explains your balanced utilization results and prevents overloading any single component.

$$W_{eff} = \frac{T_{ideal}}{T_{actual}} \in (0,1] \qquad (9)$$

Measures how close execution was to the ideal. Gains in W_eff reflect improved orchestration and memory reuse over time (as your time-per-task drops).

$$TCR = \frac{N_{success}}{N_{total}} \qquad (10)$$

The simplest but essential metric. It anchors the objective and validates the gains from coordinated agents and persistent memory.

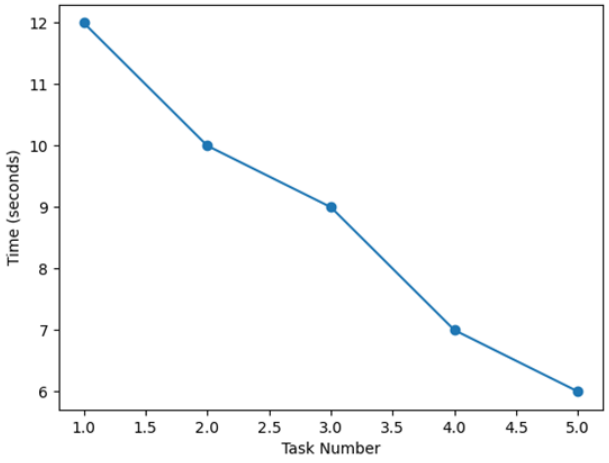

FIGURE 6. Execution Time Efficiency Trend

Figure 6 demonstrates the relationship between the number of tasks and execution time, which was decreasing with the increase in the number of tasks. The initial execution times are relatively long (12 seconds for the first task) due to the initialization of the system, the establishment of context, and the loading of memory. Over time, as more tasks are executed, the execution time drops, suggesting that the system learns and optimizes over time. This was likely due to the reuse of previously set up context and intermediate outcomes, eliminating the need for repeated calculations.

The improvement from 12 seconds to 9 seconds in the first three tasks was indicative of better coordination and learning. Through communication and context sharing, agents are more coordinated in their task execution, allowing for quicker completion. This progress indicates that the system was learning and gaining insights, allowing for quicker and more efficient processing of tasks. The drop-in execution time shows how the system becomes more efficient with time as a result of structured communication and contextual information sharing.

There was a more pronounced reduction from tasks three to four, with execution time falling from 9 seconds to 7 seconds. This dramatic reduction suggests a point at which the system achieves a state of greater optimization, presumably through effective knowledge reuse and improved task decomposition. At this point, the system likely performs minimal unnecessary work and efficient resource allocation, enabling faster task execution. This highlights the role of persistent memory and efficient access to knowledge in speeding up tasks.

In the last phase, the time to complete a task levels off at approximately 6 seconds, indicating that the system has likely achieved peak efficiency under these circumstances. This suggests that it has achieved reliable and efficient task execution without major time improvements. The pattern shows that the system not only learns but also stabilizes in terms of performance once an optimal state was reached. This trend confirms the architectural design effectively supports scalable and adaptive task execution and was therefore suitable for efficiently managing complex tasks.

Table 4: Execution Time vs Tasks

Task Number	Time (seconds)
1	12
2	10
3	9
4	7
5	6

Table 4's downward trend in execution time for tasks showcases the adaptability of the system, in which the first task took longer to execute because of the initialization and context establishment, while the following tasks benefited from experience and coordinated interactions between agents, resulting in shorter execution time; the dramatic decrease in

later tasks highlights the reuse of context and the improvement in workflow efficiency, and the convergence in lower execution times demonstrates that the system has reached an optimal state where it can execute tasks efficiently and consistently.

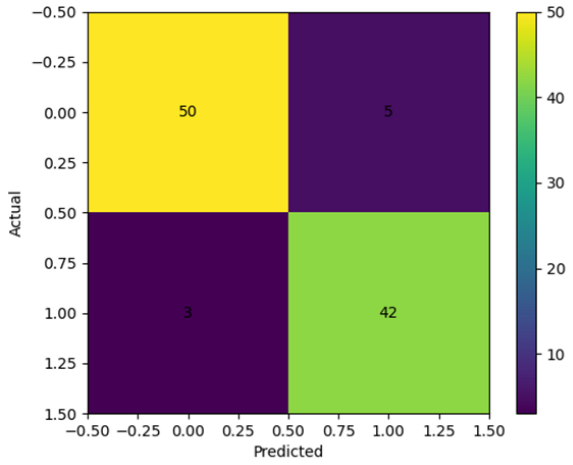

Figure 7. Classification Performance Matrix

The figure 7 offers a comprehensive analysis of the system's predictive capability through the comparison of the true results with the predictions. The confusion matrix demonstrates that the system successfully predicted a large number of cases, indicated by 50 true positives and 42 true negatives. This suggests that the system showed good performance in classifying positive and negative instances. This greater density of correct classifications suggests a well-functioning decision-making process and that the mechanisms for context awareness and execution of tasks played a role in ensuring accurate predictions.

The 5 false positives and 3 false negatives suggest that, compared to the number of correct predictions, the number of incorrect predictions was relatively small. False positives occur when the system erroneously detected a condition, whereas false negatives occur when the system failed to detect a condition. The fewer false negatives are significant as they indicate that the system had a high sensitivity and was less likely to miss important cases. This proportion of false positives and false negatives shows that the system struck a good compromise between precision and recall.

The skew of the matrix towards higher values indicates the system's reliability across different cases. The dominance of correct classifications over the number of errors indicates that the system could effectively adapt to different scenarios. This was likely due to effective use of context and structured information processing, leading to better predictions. Further, the low number of errors suggests that the system remained robust even with challenging instances.

In summary, the confusion matrix demonstrates the system's high classification accuracy and reliability. The high accuracy for both the positive and negative classes indicates the model has a well-rounded predictive capability, which was crucial for real-world applications. The low error rates also confirm the benefits of the hybrid approach for enhancing the decision-making process. These findings confirm the system's capacity to provide reliable and accurate results, underpinning its potential use in practical applications that demand accurate and consistent predictions.

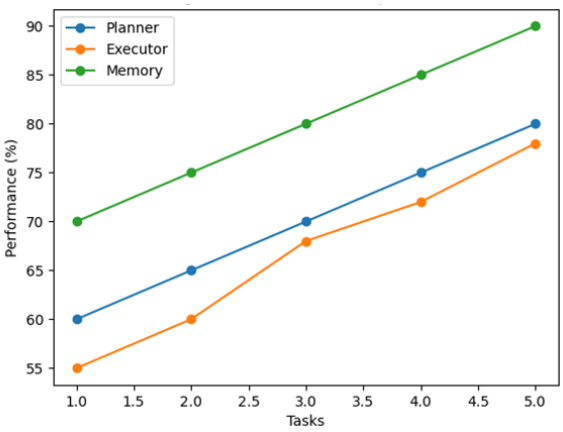

Figure 8. Agent Performance Comparison

Figure 8 illustrates a comparative performance graph of three functional agents (planner, executor, and memory) over a series of tasks. The performance of all three agents shows an upward trend, suggesting continuous improvement with increasing task difficulty. The planner starts from a middle ground and shows improvement, indicating improvements in task breakdown and planning strategies. The executor exhibits a similar pattern but with a lower initial performance, indicating that the execution processes needed further adaptation. On the other hand, the memory agent consistently exhibits the highest performance level, showcasing its importance for the performance.

The steady rise in the performance of the planner agent from 60% to 80% implies improved task planning and execution. Over time, the planner agent takes advantage of the growing understanding of the task environment, allowing it to make better decisions. This growth indicates that the system effectively uses previous knowledge to enhance planning approaches, eliminating uncertainties and enhancing agent coordination. The continuous improvement also suggests that planning strategies are effective in adapting to task complexities.

There's a significant improvement in the executor agent from 55% to 78%, indicating better task execution performance. At the beginning, the sub-optimal performance could be explained by difficulties in managing task execution-level complexities, such as tool use and interactions with other agents. But as the tasks progress, the executor receives better guidance from the planner and has access to more information. This results in faster and more accurate execution and improved goal alignment. The approach of the executor's performance towards the planner's performance suggests growing coordination between the planning and execution

phases.

The memory agent continues to perform better than other agents, from 70% to 90%, which speaks to the power of persistent memory to improve system performance. This was due to the effective storage, recall, and reutilization of contextual information, facilitating effective operation of other agents. The upward trend suggests that using memory becomes more useful during the course of the task, enabling quicker decision-making and avoiding duplication. In summary, the graph shows that the improvement in all the agents' performance leads to more efficient system performance, and memory plays an important role in improving performance and enabling collaboration among agents.

Table 5. Agent Performance Comparison

Task	Planner (%)	Executor (%)	Memory (%)
1	60	55	70
2	65	60	75
3	70	68	80
4	75	72	85
5	80	78	90

Table 5 shows the continuous improvement across all agents, where the planner gradually improved performance in terms of task planning and coordination, the executor performed better over time due to improved guidance and context information, and the memory agent always performed better than others because it provided information about the context; the upward trends indicate increased coordination among agents and improved collaboration, demonstrating that persistent memory played a significant role in improving planning and execution processes and hence system performance and efficiency.

Conclusion

This research developed an integrated framework of protocol-based (standardized) communication, multi-agent collaboration, persistent memory, and workflow orchestration to resolve existing issues in designing intelligent systems. The addition of a structured communication layer facilitated communication between agents and external tools, and the shared context mechanism contributed to consistency and coherence in task execution. The persistent memory element significantly contributed to the system's intelligence, allowing effective storage, retrieval, and reuse of contextual information and avoiding unnecessary computation, ensuring decision continuity.

Empirical analysis showed significant gains in various performance metrics such as task success rate, execution time, response latency, and resource consumption. The novel approach significantly improved the performance of traditional single-agent and stateless multi-agent systems by enabling collaborative task decomposition and dynamic execution. The resulting decrease in execution time across multiple tasks demonstrated the benefits of memory-based optimization, while the even distribution of resource utilization across different tasks validated the scalability and efficiency of the system in managing complex tasks. The correlation between memory usage and task execution time confirmed the value of incorporating retrieval mechanisms in agent designs.

The findings also suggested that effective agent coordination, enhanced by persistent context management, enhanced system reliability and stability. Adaptability to changing task demands and reuse of past experience allowed the system to consistently deliver good performance across different operating scenarios. The modular design provided flexibility and enabled the framework to be easily extended or integrated with various applications with minimal modification.

Finally, the framework offers a scalable and efficient approach to future intelligent systems by integrating protocol standardization, memory-based reasoning, and multi-agent coordination. The research helps advance automated workflow orchestration and demonstrates the value of integrating contextual communication and persistent memory for the development of adaptive and efficient AI systems.

Data Availability Statement

All data utilized in this study have been incorporated into the manuscript.

Authors' Note

The authors declare that there is no conflict of interest regarding the publication of this article. Authors confirmed that the paper was free of plagiarism.

References

[1] Krishnan, N. (2025). Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. arXiv preprint arXiv:2504.21030.

[2] Venkiteela, P. (2025). The New Interoperability Paradigm Model Context Protocol (MCP), APIs, and the Future of Agentic AI. Comput. Fraud Sec, 8(1), 1259-1271.

[3] Kumar, N., Sagar, V., & Jain, G. (2025, November). A Comprehensive Study and Implementation of Agentic AI via MCP Servers. In 2025 International Conference on Emerging Technologies and Innovation for Sustainability (EmergIN) (pp. 42-49). IEEE.

[4] Ehtesham, A., Singh, A., Gupta, G. K., & Kumar, S. (2025). A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). arXiv preprint

arXiv:2505.02279.
[5] Ray, P. P. (2025). A survey on model context protocol: Architecture, state-of-the-art, challenges and future directions. Authorea Preprints.
[6] Yang, Y., Chai, H., Song, Y., Qi, S., Wen, M., Li, N., ... & Zhang, W. (2025). A survey of ai agent protocols. arXiv preprint arXiv:2504.16736.
[7] Lei, Y., Xu, J., Liang, C. X., Bi, Z., Li, X., Zhang, D., ... & Yu, Z. (2025). A Comprehensive Survey on Model Context Protocol: Architecture, Tool Integration, and the Future of AI Interoperability. Tool Integration, and the Future of AI Interoperability (December 23, 2025).
[8] Murthya, S. L., Selvama, S. P., & Welßa, M. (2025). Multi-Agentic Workflows and Long-Term Memory Use Cases in AI-Builder.
[9] Singh, A., Ehtesham, A., Kumar, S., & Khoei, T. T. (2025). A survey of the model context protocol (mcp): Standardizing context to enhance large language models (llms).
[10] Gabbita, B. S. T. (2025). Towards secure agentic workflows: a MAESTRO-based assessment framework for Model Context Protocol and Agent-to-Agent Protocol: a thesis in Computer Science (Doctoral dissertation, University of Massachusetts Dartmouth).
[11] Савельєв, Р. І., & Дендюк, М. В. (2025). Leveraging the model context protocol architecture for data analytics in software distributed systems. Scientific Bulletin of UNFU, 35(6), 49-59.
[12] Ray, P. P. (2025). A review on agent-to-agent protocol: Concept, state-of-the-art, challenges and future directions. Authorea Preprints.
[13] Yu, C., Cheng, Z., Cui, H., Gao, Y., Luo, Z., Wang, Y., ... & Zhao, Y. (2025, May). A survey on agent workflow–status and future. In 2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD) (pp. 770-781). IEEE.
[14] Errico, H., Ngiam, J., & Sojan, S. (2025). Securing the Model Context Protocol (MCP): Risks, Controls, and Governance. arXiv preprint arXiv:2511.20920.
[15] Koubaa, A. (2025). Agent Operating Systems (Agent-OS): A Blueprint Architecture for Real-Time, Secure, and Scalable AI Agents. Authorea Preprints.
[16] Liao, C. C., Liao, D., & Gadiraju, S. S. (2025, November). Agentmaster: A multi-agent conversational framework using a2a and mcp protocols for multimodal information retrieval and analysis. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (pp. 52-72).
[17] Hasan, M. M., Li, H., Fallahzadeh, E., Rajbahadur, G. K., Adams, B., & Hassan, A. E. (2025). Model context protocol (mcp) at first glance: Studying the security and maintainability of mcp servers. arXiv preprint arXiv:2506.13538.
[18] Styer, M., Patlolla, K., Mohan, M., & Diaz, S. (2025). Agent Tools & Interoperability with MCP. Kaggle.[Online]. Available: https://www. kaggle. com/whitepaper-agent-tools-andinteroperability-with-mcp.[Accessed: Nov. 13, 2025].
[19] Oyelayo, O., Abedu, S., Khatoonabadi, S., & Shihab, E. (2025). Developer Challenges in the Adoption of Model Context Protocol in AI Agent Development.
[20] Dinh, T. (2025). LLM-Based Multi-Agent Architecture for Transport Network Automation Using MCP and A2A Protocols.
[21] Dibia, V. (2025). Designing Multi-Agent Systems: Principles, Patterns and Implementation for AI Agents. Victor Dibia.
[22] Liao, C. C., Liao, D., & Gadiraju, S. S. AgentMaster: A Modular Multi-Agent Framework with A2A and MCP Protocols via a Unified Conversational Interface. In NeurIPS 2025 Workshop on Bridging Language, Agent, and World Models for Reasoning and Planning.
[23] Derouiche, H., Brahmi, Z., & Mazeni, H. (2025). Agentic ai frameworks: Architectures, protocols, and design challenges. arXiv preprint arXiv:2508.10146.
[24] So, R. (2025). MCP Servers for Scientific Workflows. arXiv preprint arXiv:2504.13438.
[25] Petrova, T., Bliznioukov, B., Puzikov, A., & State, R. (2025). From Semantic Web and MAS to Agentic AI: A Unified Narrative of the Web of Agents. arXiv preprint arXiv:2507.10644.
[26] Huang, K., & Hughes, C. (2025). Agentic AI Communication Protocols and Security. In Securing AI Agents: Foundations, Frameworks, and Real-World Deployment (pp. 81-110). Cham: Springer Nature Switzerland.
[27] Ersoy, P., & Erşahin, M. (2025, October). Protocol-Driven Agentic Integration of LLMs: A Multi-tool Use Case. In International Joint Conference on Computational Intelligence (pp. 139-148). Cham: Springer Nature Switzerland.
[28] Sarkar, A., & Sarkar, S. (2025). Survey of llm agent communication with mcp: A software design pattern centric review. arXiv preprint arXiv:2506.05364.
[29] Singh, A., Ehtesham, A., Lambe, M., Grogan, J., Singh, A., Kumar, S., ... & Raskar, R. (2025, November). Evolution of ai agent registry solutions: Centralized, enterprise, and distributed approaches. In 2025 IEEE 7th International Conference on Cognitive Machine Intelligence (CogMI) (pp. 507-515). IEEE.
[30] Zhang, W., Zeng, L., Xiao, Y., Li, Y., Zhao, Y., Cui, C., ... & An, B. (2025). AgentOrchestra: Orchestrating hierarchical multi-agent intelligence with the Tool-Environment-Agent (TEA) protocol.
[31] Tiwari, A., Bhalla, A., & Prasad, D. G. Model Context Protocol for Vision Agents: Schema, Memory, and World Model Implications. In NeurIPS 2025 Workshop on Bridging Language, Agent, and World Models for Reasoning and Planning.
[32] Matsumoto, D., Watarai, K., Okada, S., & Mitsunaga, T. (2025, October). A2A Routing Service with MCP Integration: Bridging Security,

Auditability, and Governance in AI Agent Systems. In 2025 IEEE International Conference on Computing (ICOCO) (pp. 60-65). IEEE.
[33] Jiang, W., & Hu, F. (2025). Artificial Intelligence Agent-Enabled Predictive Maintenance: Conceptual Proposal and Basic Framework. Computers, 14(8), 329.
[34] Ercole, D. (2025). AI Agents Leveraging RAG and MCP for Insurance Knowledge Management and Enterprise Workflow Automation (Doctoral dissertation, Politecnico di Torino).
[35] Vinay, P. T., & Saurav, S. K. (2025, December). CoMAS-HPC: A Collaborative Multi-Agent System for HPC Administration. In 2025 Supercomputing India (SCI) (pp. 1-8). IEEE.
[36] Pham, M. T. (2025). Development of NLP Interactive Database Query System using Large Language Models Combined with AI Agent (Doctoral dissertation, Vietnam-Korea University of Information and Communication Technology).
[37] Yang, M., Lovett, N., Li, B., & Hou, Z. (2025). Towards dynamic learner state: Orchestrating ai agents and workplace performance via the model context protocol. Education Sciences, 15(8), 1004.
[38] Gaire, S., Gyawali, S., Mishra, S., Niroula, S., Thakur, D., & Yadav, U. (2025). Systematization of knowledge: Security and safety in the Model Context Protocol ecosystem. arXiv preprint arXiv:2512.08290.
[39] Tiwari, A., Bhalla, A., & Prasad, D. (2025). Model Context Protocol for Vision Systems: Audit, Security, and Protocol Extensions. arXiv preprint arXiv:2509.22814.
[40] Rohitha, K. P. (2025). Agentic AI-Building Autonomous Intelligent Systems for Enterprise Applications. AGPH Books| AG Publishing House.

Embargo period: The article has no embargo period.

To cite this Article: To cite this Article: B. Jyothi, Bujula NehaSri, Gongidi Jayadeep, and Ananthula Akshitha, Model Context Protocol (MCP) Server for Multi-Agent AI Assistants with Persistent Memory: An Integrated Framework for Automated Workflow Management, Engineering Research 1.1(2025):1-2. https://doi.org/10.71443/er.ar17