Evaluating Multi-Agent Coordination Frameworks: A Comparative Study of Cerwai, AutoGen, LangGraph, and MCP

Chaitanya Palta[1], Dr. Meenu Kaushik[2], Dr. Akhilesh Das Gupta[3]
[1, 2, 3] Department of Artificial Intelligence and Data Science
[1, 2, 3]Institute of Professional Studies Delhi, India

Abstract- *This paper presents a comparative study of four major approaches currently shaping Multi-Agent Systems (MAS): the structured, graph-based model of LangGraph, the decentralized conversational setup of AutoGen, the role-based orchestration used in CrewAI (or Cerwai), and the foundational security and architecture layer provided by the Model Context Protocol (MCP). The goal is to clearly outline their architectural differences, highlight real-world performance insights, and ex- plain the underlying engineering trade-offs so that developers and researchers can make informed decisions when building production-ready agentic AI systems.*
Architecturally, each framework brings a distinct approach. LangGraph's Directed Acyclic Graphs (DAGs) enable consistent state management and deterministic task execution, making it ideal for complex, long-running workflows with checkpointing. AutoGen, by contrast, focuses on agility and conversational simplicity, excelling in interactive applications that require quick coordination between agents. CrewAI streamlines the creation of autonomous teams through predefined roles and task structures, offering an approachable entry point for specialized automation. A key takeaway is that the Model Context Protocol (MCP) is not a competing framework but a foundational standard for security, tool management, and interoperability across MAS ecosystems. It mitigates major risks related to arbitrary tool execution and enables safe, standardized integration between different agentic frameworks. Based on performance metrics like Response Consistency Index (RCI) and latency, the study concludes that there is no universally superior framework—each is suited to different goals. The right choice depends on whether the priority is maximizing reliability and accuracy (high RCI) or achieving low-latency responsiveness for interactive systems.

Keywords- Multi-Agent Systems, Agentic AI, Framework Evaluation, AutoGen, LangGraph, Model Context Protocol, Co- ordination, Large Language Models (LLMs), Tool Use

I. INTRODUCTION

A. The Rise of Multi-Agent Systems

Artificial intelligence is undergoing a major shift from single, monolithic models to decentralized, collaborative net- works known as Multi-Agent Systems (MAS). MAS consists of multiple autonomous agents that interact within a shared environment to solve complex problems that a single system cannot handle efficiently. This approach relies on distributed control and specialized agent roles, allowing collaboration, coordination, or competition to achieve shared objectives. As a result, MAS enhances accuracy, adaptability, and scalability, enabling AI to move from basic text generation to complex, multi-stage reasoning and execution. By supporting dynamic interaction and contextual decision-making, MAS overcome the limitations of traditional Sequential One-Agent Pipelines (SOP), such as semantic drift and incomplete reasoning. Through specialized teams of agents—like coders, planners, and evaluators—MAS achieve iterative refinement, leading to stronger and more reliable performance across complex benchmarks.

B. Problem Statement: The Fragmentation of Coordination Frameworks

The rapid rise of agentic AI has led to a fragmented landscape of coordination frameworks, each offering its own approach to task delegation, memory handling, and control flow. Frameworks like LangGraph, AutoGen, and CrewAI differ greatly in design philosophy and implementation. This diversity, while innovative, has created a challenge for developers and system architects—there is no common standard or clear empirical basis for comparison. As a result, many developers select frameworks based on short-term convenience rather than long-term scalability, often facing costly redesigns later when their systems need higher consistency, flexibility, or interoperability than the chosen framework can provide.

C. Introducing the Subjects of Study

This research focuses on three major coordination frameworks and one essential protocol standard:

1) LangGraph: A graph-based framework built upon LangChain, utilizing Directed Acyclic Graphs (DAGs). Each node represents an agent task, and edges define the control flow. It is defined by its ability to support dynamic transitions, loops, and centralized, persistent state management.
2) AutoGen: Developed by Microsoft, this conversation-centric framework models tasks as asynchronous dialogues between agents. It provides a generic and extensible design that enables seamless, multi-turn interactions using both natural language and code.
3) CrewAI: An open-source, lean, Python-native framework focusing on high-autonomy teams, specialized roles, and delegated tasks. It facilitates collaboration by assembling agents into crews to execute common goals, often employing predefined templates for agent collaboration.
4) Model Context Protocol (MCP): An open protocol developed primarily by Anthropic and adopted by major providers like OpenAI and Google DeepMind. MCP utilizes JSON-RPC 2.0 messages to standardize how applications provide contextual information, expose tools, and build composable workflows for LLMs.

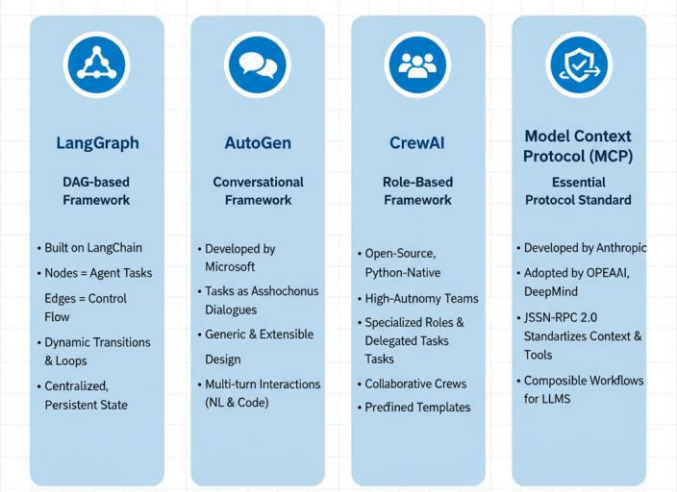

Fig. 1. MAS Coordination Framework Comparison

II. LITERATURE REVIEW

The development of Multi-Agent Systems (MAS) stems from the limitations of single-agent architectures. Single-agent frameworks depend on one language model to handle a wide range of tasks, which often leads to performance bottlenecks and difficulties maintaining context across multiple stages. Systems like the Sequential One-Agent Pipeline (SOP) fre- quently experience semantic drift, limited internal verification, and weak adversarial testing. As a result, their final outputs tend to be inconsistent or suboptimal, especially in complex, multi-step or algorithmic problem-solving scenarios.

A. LangGraph's DAG Framework

LangGraph structures multi-agent coordination through di- rected acyclic graphs (DAGs), providing a clear and control- lable flow of execution. Each node corresponds to a specific agent or task, while the edges define how information and control move between them, supporting loops, branching, and dynamic behavior based on state changes. Grounded in graph theory, this approach enables developers to design complex, transparent workflows with strong control and visualization. LangGraph uses a centralized, mutable State object that flows through all nodes, allowing agents to update and reference shared context for consistent decision-making. This archi- tecture also supports checkpointing, ensuring reliability and smooth recovery in large-scale or long-running systems.

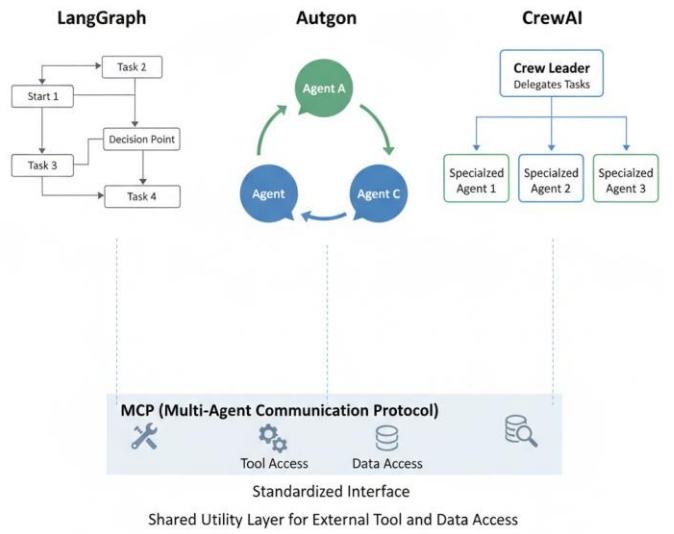

Fig. 2. Conceptual Diagram: MAS Coordination Framework.

B. AutoGen's Conversational Framework

AutoGen uses a message-passing, conversational, and event- driven approach. It models tasks as dialogues between agents, where each agent listens for messages, processes them, and responds asynchronously. This design makes it easier to build complex LLM-based applications by following a conversation- focused programming style that adapts well to tasks like solving math problems, writing code, and making decisions. AutoGen focuses on conversational agent

architecture, encour- aging natural language interactions and flexible role-playing. Unlike LangGraph, it gives each agent its own memory cache, so agents maintain local context. Shared information is exchanged through explicit messages or a common datastore. While this method improves modularity and is easier for beginners to use, keeping a consistent global state requires extra coordination logic.

C. CrewAI's Role-Based Model

CrewAI is a fast and lightweight Python framework de- signed to improve autonomy and teamwork among agents through well-defined roles and assigned tasks. It simplifies the creation of specialized agents—called a "crew"—that work together to complete tasks, often within customizable work- flows or in connection with third-party tools. CrewAI focuses on role-based coordination, making it especially useful in situations where agents need to collaborate based on hierarchy and specific expertise. It uses structured, role-based memory with RAG support, ensuring that each agent's behavior stays context-aware. The main advantage of CrewAI is its simplicity and focus on team execution, allowing users to easily define agents using YAML-based configuration.

III. METHODOLOGY

A. Evaluation Framework

We define three key evaluation dimensions:

TABLE I
EVALUATION DIMENSIONS AND METRICS

Metric	Description	Example Measure
Autonomy	Ability to act with mini- mal human input.	Steps per human in- tervention
Coordination Efficiency	Quality of multi-agent communication and synchronization.	Task completion time, conflict rate
Reasoning Consistency	Logical coherence and output reliability across iterations.	Agreement score, Response Consistency Index (RCI)

B. Experimental Task Definition: Complex Research Synthesis

To effectively evaluate coordination, state management, and resource efficiency, the selected experimental task is a complex research synthesis workflow. This task naturally involves multiple stages and requires iterative improvement, including: Data Gathering: Using external tools such as file systems or APIs, integrated through the MCP standard. Analysis and Critique: Employing a multi- agent feedback loop, such as a peer review or critic mechanism, to evaluate intermediate results. Refinement: Repeating or negotiating steps dynamically until a desired quality level is reached. Final Writing: Combining and organizing the findings into a clear, well-structured output. This workflow is well-suited for testing LangGraph's precise task sequencing, AutoGen's capability for deep multi-turn interactions, and CrewAI's ability to handle high-autonomy role execution.

TABLE II
BENCHMARK TASKS AND THEIR EVALUATION FOCUS

Benchmark Task	Primary Focus	Key Challenge Tested
Collaborative Code Generation	Two agents jointly write a Python function using an API specification.	Role specialization, feedback mechanisms
API Orchestration	Agents interact with exter- nal APIs (e.g., weather, fi- nance) to complete a com- plex goal.	Tool integration, sequential depen- dency handling
Research Synthesis (Iterative)	Agents collectively sum- marize and verify lit- erature sources through multi-step refinement.	State management, dynamic looping, critique
Negotiation/Debate Task (CVCP)	Agents argue opposing views and converge to consensus or optimal solu- tion.	Conflict resolution, adversarial testing, voting resolution

C. Simulation Environment and Tools

The simulation environment used a shared LLM instance for all agent requests to simulate real-world resource contention.

This setup is important because, in practical scenarios, multi- ple agents often access the same LLM, which can cause heavy loads and latency between agents. Testing under these shared conditions helps evaluate how efficiently each framework manages queuing and minimizes overhead.

Tool integration was standardized through the Model Con- text Protocol (MCP). A required research tool was provided via a dedicated MCP Server, functioning like a univer- sal "USB-C port" for AI applications. This setup ensured that all three frameworks—LangGraph, AutoGen, and Cre- wAI—accessed the tool through the same standardized JSON- RPC 2.0 interface. This eliminated differences caused by tool implementation and allowed precise measurement of the latency overhead introduced by the protocol layer itself.
To maintain quantitative accuracy, an observability pipeline was implemented. It included advanced logging and tracing tools designed for multi-agent systems, capturing detailed Agent Interaction Logs that record reasoning, actions, and

communications. These logs were later used for RCI (Reasoning Chain Integrity) and efficiency analysis.

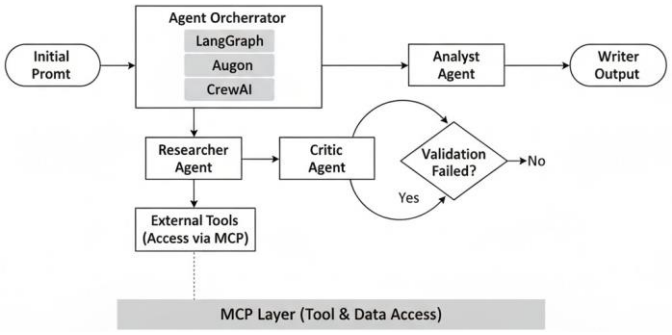

Fig. 3. Experimental Workflow Diagram

D. Quantification of Metrics

1) **Latency Measurement:** End-to-end latency was mea- sured from prompt submission to final output, along with inter-agent latency—the time between receiving a message or state update and generating the next response. These metrics reflect real-world conditions for optimizing multi-agent performance.
2) **RCI Calculation:** The Response Consistency Index (RCI) was computed over 50 runs of the research synthesis task using identical inputs under controlled memory limits. Output differences were evaluated using semantic distance metrics, where higher RCI values in- dicate greater consistency and robustness against noise.
3) **Cost Analysis:** Token usage was tracked for each suc- cessful run, separating prompt and completion tokens. This allowed a comparative cost-per-request study to understand the operational efficiency of each framework.

The table below outlines the experimental setup and key metrics for each evaluated framework.

TABLE III
FRAMEWORK BENCHMARK TESTS AND EVALUATION METRICS

Framework	Test Focus	Key Metric
LangGraph	Coordination Test — Looping/Branching (Critic Agent)	State Consistency Test — Centralized state updates under load
AutoGen	Group Chat / Negotiation Test	Message passing & local cache synchro- nization
CrewAI	Role delegation & Task queuing Test	Structured memory effectiveness (RAG integration)
MCP	Tool and Context expo- sure layer Test	N/A (Protocol)

IV. EXPERIMENTAL SETUP

A. LangGraph Implementation Details

The LangGraph implementation used a TypedDict called AgentState to maintain centralized control over contextual information such as messages, current step, and iteration count. The workflow followed a directed graph where the research node gathered data using the MCP tool, followed by the analysis node. A critic node introduced a feedback loop if the analysis was insufficient, the process looped back to the research or analysis stage for refinement before reaching the writer node.

This graph-based structure supports stateful transitions and efficient checkpointing, ensuring smooth workflow continuity and resource optimization. Its explicit flow control makes it ideal for complex tasks that require consistent validation and accuracy.

B. AutoGen Implementation Details

AutoGen was set up with a user-proxy agent that interacted with specialized assistant agents—Researcher, Analyst, and Critic. Coordination occurred through natural language dia- logue with clear message termination rules, enabling agents to discuss, negotiate, and refine the research task through multi- turn interactions.

A major focus was state synchronization, as each AutoGen agent maintains its own memory cache. To ensure a shared understanding, key findings were explicitly passed between agents using structured messages. Additionally, sandboxing was enabled to securely execute any generated Python code for data processing while maintaining safety standards.

C. CrewAI Implementation Details

The CrewAI setup included a Crew of three agents: Re- searcher (for data gathering), Analyst (for interpretation), and Writer (for synthesis). Each agent was assigned a defined Role with specific Tasks arranged in sequence. CrewAI's design focuses on simplicity, with the framework handling orchestration automatically based on assigned roles and goals. Its role-based memory with RAG integration enabled context-aware behavior. The workflow followed a task pipeline, where completing one agent's task automatically triggered the next, ensuring smooth and efficient collaboration.

D. MCP Integration Setup

The Model Context Protocol (MCP) acted as a unified abstraction layer for all external tools. A dedicated MCP Server hosted essential utilities like file access and web search, handling JSON-RPC 2.0 communication. This provided Lang- Graph, AutoGen, and CrewAI with a standardized interface to list available tools and dynamically fetch contextual prompts. MCP's standardization allowed the research synthesis task to replace data-gathering tools without changing the core agent logic, encouraging component reuse and long-term scalability. The experiment also examined performance under resource contention, where agents shared a single external LLM. In such cases, system bottlenecks depend on LLM serving perfor- mance. It was expected that CrewAI's lightweight design and AutoGen's low decision overhead would show better queuing and lower inter-agent latency compared to LangGraph, which incurs higher overhead due to its complex, centralized state
updates across multiple nodes.

V. RESULT AND ANALYSIS

A. Coordination Mechanism Effectiveness

The analysis showed that coordination efficiency largely depends on task structure. AutoGen performed well in short, interactive loops like clarifying data points, and its conversa- tional design made it ideal for human-in-the-loop workflows. However, deeper conversations led to higher token use if not limited.

In contrast, LangGraph maintained strong sequential effi- ciency through its clear graph structure, avoiding random loops but requiring careful setup—such as adding a Critic Agent—to achieve dynamic feedback. CrewAI offered predictable task flow and clear role-based execution, focusing on fast, parallel completion of specialized sub-tasks.

B. State Management and Consistency Analysis (RCI)

The Response Consistency Index (RCI) results highlighted clear differences between the frameworks.

LangGraph achieved the lowest RCI (highest consistency) across 50 runs. Its centralized AgentState and strong check- pointing maintained global context, reducing semantic drift during repeated or looping steps. This makes it highly reliable for deterministic tasks like financial reporting or compliance checks.

AutoGen showed higher RCI (lower consistency). Its per- agent memory promotes modularity but requires extra co-ordination logic to maintain shared context. Inconsistencies often arose when agents relied on outdated or incomplete local memory, leading to more variable outputs under noisy or constrained conditions.

CrewAI delivered moderate RCI results. Its structured, role- based memory supported solid internal consistency, but its reliance on RAG-based context retrieval made performance sensitive to retrieval quality and latency.

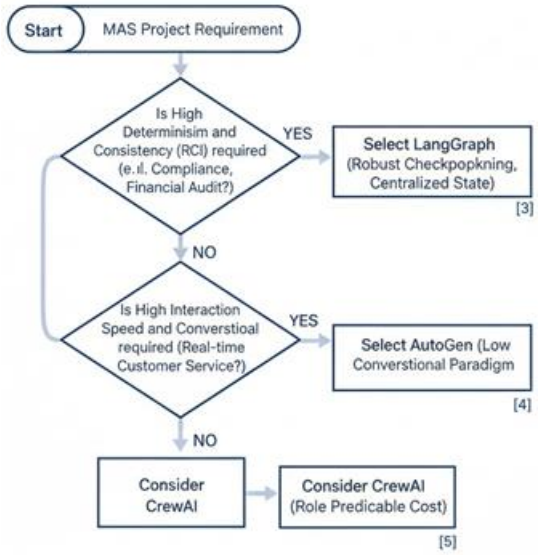

Fig. 4. Optimal Framework Selection based on RCI and Latency

C. Performance Metrics: Latency and Token Efficiency

1) Latency Analysis: Both AutoGen and CrewAI showed lower latency per agent decision compared to Lang-Graph. CrewAI's lightweight Python core and Auto-Gen's minimal design allowed smoother queuing and faster responses when sharing the same LLM, thanks to their simpler context handling. LangGraph, while slightly slower due to managing a centralized state object, trades speed for stronger reliability and state in- tegrity—essential for complex, long-running workflows. Similar to systems like Kairos, these results highlight how internal scheduling strategies directly affect real- world performance.
2) Token Cost Analysis: In terms of token usage, Lang-Graph's centralized state can increase prompt size if context objects are not efficiently managed. AutoGen's conversational approach risks high token consumption through repeated negotiation loops, while CrewAI's structured, role-based design maintained the most stable and predictable cost pattern. Regardless of framework, optimizing cost requires context compression and a hi-erarchical memory system that separates short, medium, and long-term storage.

VI. DISCUSSION

A. Architectural Trade-Offs: Control vs. Agility

The comparative analysis highlights a key trade-off in MAS frameworks—control versus agility.

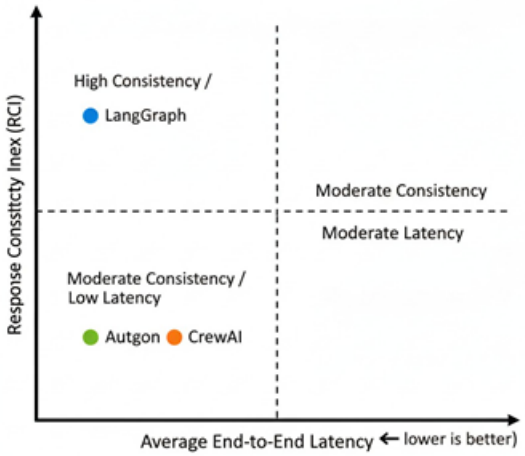

Fig. 5. Comparative Performance: Consistency vs. Latency Trade-off.

LangGraph excels in precision and structured control. Its graph-based model supports clear branching, looping, and checkpointing, allowing the system to retain context across complex workflows. This makes it ideal for long-running, deterministic tasks that demand reliability and orchestration. However, it comes with a steeper learning curve, requiring familiarity with graph theory and state management, and its tight integration with LangChain can reduce flexibility.

AutoGen, on the other hand, focuses on agility and conver- sational flow. It coordinates agents through dialogue, enabling fast prototyping and adaptability. Its AutoGen Studio offers an easy, no-code interface for experimentation. While well- suited for dynamic, interaction-heavy tasks, its decentralized memory model requires strong programming logic to maintain consistent state across agents in structured workflows.

B. The Strategic Role of CrewAI

CrewAI offers a balanced middle ground, emphasizing ease of setup and fast deployment. With its clear role definitions and YAML-based configuration, it provides an accessible entry point for building specialized, autonomous teams. Its main strength lies in simplifying collaboration between AI agents—and even human participants—across various tasks. The modular design also supports scalable, parallel execution. However, as workflows grow more complex and non-linear, CrewAI's role-based structure can feel limiting, often requiring a shift to a more flexible system like LangGraph for finer control.

C. Protocol Standardisation and Interoperability (MCP)

The Model Context Protocol (MCP) plays a vital role in solving inefficiencies found in today's diverse agentic ecosys- tems. By standardizing how tools and context are shared, it acts as the central control plane for agentic AI.

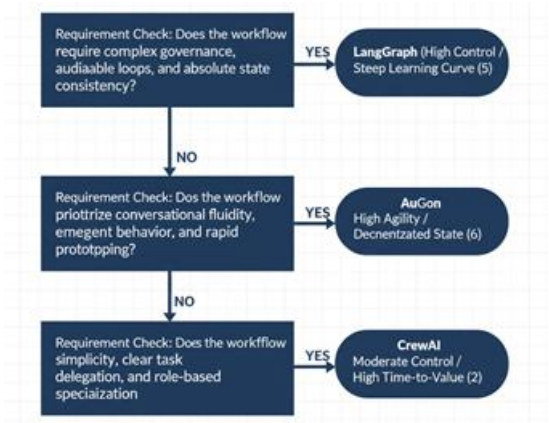

Fig. 6. Mapping Architectural Needs to Framework

This standardization allows for easier reuse and better scalability. Instead of creating custom integrations for each agent-tool pair across different MAS frameworks, a single MCP server can handle all requests. This reduces infrastructure costs and speeds up deployment, as organizations can connect their existing systems through MCP without major rewrites, making them instantly usable by AI agents.

D. Engineering Complexities: Debugging and Observability

A key challenge in multi-agent coordination is the new level of debugging complexity that traditional methods cannot address. Multi-agent systems often produce non-deterministic and emergent behavior, making it hard to trace errors across agents. When failures occur, causes are distributed, stack traces are unhelpful, and reproducing issues becomes difficult. Track- ing planning loops and linking scattered logs is especially challenging, a problem known as "Attribution Difficulty," where even advanced LLMs struggle to pinpoint the exact failure step.

To handle this, specialized Multi-Agent Evaluation Systems (MAES) and observability tools like Eval360 are essential for reliable deployment. While LangGraph provides some visualization through its graph model, all frameworks require detailed Agent Interaction Logs to trace reliability,

consistency, and context flow—evaluating every step, not just the final result.

E. Scalability and Latency Management

Scaling multi-agent systems is not only about adding more computing power but also about improving orchestration and reducing latency. Since LLM serving often becomes the main bottleneck, the framework's internal design heavily affects performance. LangGraph scales well through asynchronous processing and distributed architecture, while AutoGen stands out for its strong automation and async support. Research on

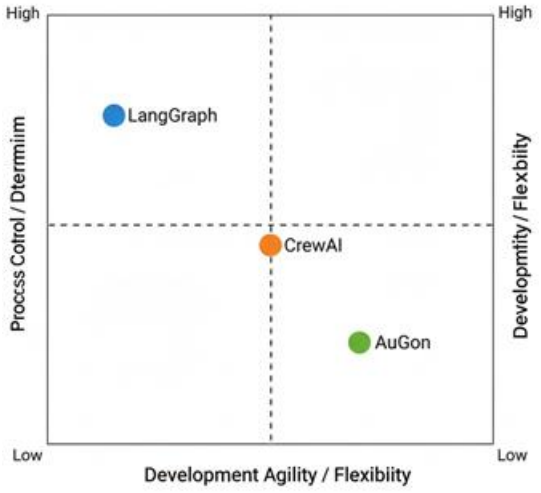

Fig. 7. Architectural Trade-Off Space: Control vs. Agility

orchestration systems like Kairos shows that efficient scheduling based on latency and memory needs is essential—making advanced internal scheduling a key factor for reliable scalability in these frameworks.

VII. CONCLUSION

The study shows that choosing a MAS coordination frame- work depends on balancing control, consistency, and devel- opment speed. LangGraph is best for high-consistency and compliance-critical systems, thanks to its centralized state and clear graph flow—ideal for financial or regulated workflows. AutoGen focuses on flexibility and rapid prototyping, making it suitable for interactive and customer-facing applications. CrewAI is the simplest to use, excelling in team-based au- tomation and role-driven tasks. The Model Context Protocol (MCP) remains crucial for secure, scalable, and interoperable deployments, ensuring standardization, user safety, and eco- nomic efficiency across frameworks.

VIII. EXTENDED APPLICATIONS

A. Decentralized Finance (DeFi) and Economic Simulation

Multi-Agent Systems (MAS) play a key role in decentral- ized finance by modeling interactions between autonomous entities like traders and lenders. LangGraph is well-suited for simulating trading and lending strategies that rely on structured, rule-based logic, while AutoGen's conversational setup can represent negotiation or dispute resolution within DAOs. Given DeFi's strict security needs, MCP's built-in consent and data protection standards are vital for safely connecting external financial data services.

B. Industrial Orchestration and Workflow Optimization

MAS are widely used to improve complex operations like supply chains, logistics, and data center management. These systems need efficient planning and resource allocation. Research like Kairos, which reduces latency through better resource scheduling for heavy LLM workloads, shows clear performance gains. Whether using LangGraph for structured control or CrewAI for parallel execution, latency-aware scheduling and optimized resource use are key for real-time industrial applications.

C. Advanced Human-Agent Collaboration

The future of AI agents lies in building systems that enable smooth collaboration between humans and MAS. This involves creating interfaces that allow flexible control and easy interaction. AutoGen's support for human-in-the-loop setups and its visual AutoGen Studio make it ideal for such collaborative systems, letting users guide or adjust workflows naturally through conversation without breaking the process flow.

IX. PRACTICAL IMPLEMENTATION CONSIDERATIONS

A. Debugging, Observability, and Non-Determinism

Deploying MAS in real environments requires strong meth- ods to handle unpredictability and emergent behavior. Since traditional debugging fails in such systems, detailed Agent Interaction Logs are essential to trace every decision and action. Using evaluation tools like Eval360 helps identify errors, ensure reliability, and overcome the "Attribution Difficulty," making such observability systems crucial for stable and scalable MAS deployment.

B. Resource Management and Cost Optimization

LLM token usage is the main driver of operational cost in MAS, making efficient resource management essential. A layered memory system—short-term for context, medium-term for recent actions, and long-term via RAG for history—helps optimize performance. Using context compression and batch processing minimizes redundant data and token use. Although building custom agent systems can be expensive, applying these optimization strategies ensures significant long-term sav- ings in compute and token costs.

C. Security and Trust Implications of the Model Context Protocol

Implementation of agentic tools, regardless of the frame- work, must comply with the security standards defined by the Model Context Protocol (MCP). Since MCP enables data access and code execution, strong consent and authorization mechanisms are essential. Key requirements include:

1) User Consent: Users must explicitly approve all data access and actions. Hosts should not share or transmit user data without clear consent.
2) Tool Safety: Tools involve arbitrary code execution and must be handled cautiously. Tool descriptions should be treated as untrusted unless verified by a secure server,

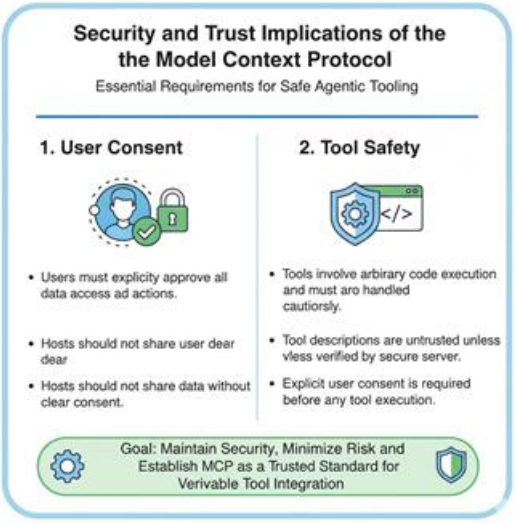

Fig. 8. Architectural Trade-Off Space: Control vs. Agility

and explicit user consent is required before any tool execution. Following these principles is essential to maintain security, minimize risk, and establish MCP as the trusted standard for safe and verifiable tool integration.

X. BENCHMARK RESULTS

The following tables summarize the key insights from the comparative architectural analysis and the quantitative performance metrics obtained from the Complex Research Synthesis task and the CVCP case study.

TABLE IV
COMPARISON OF FRAMEWORKS AND PROTOCOLS

Framework	Coordination Paradigm / State Model	Primary Use Case
LangGraph	Graph-based (DAGs, Looping); centralized, mutable, checkpointed state	Complex, stateful, non-linear workflows
AutoGen	Conversational / event-driven; decentralized per-agent caches	Dialogue-driven, rapid prototyping
CrewAI	Role-based orchestration; structured role memory	Task delegation, multi-agent collaboration
MCP	Open context/tooling protocol; standardized interface (stateless)	Tool integration, security, interoperability

TABLE V
FRAMEWORK CONSIDERATIONS AND DEPLOYMENT CHALLENGES

Consideration	Framework Insights	Deployment Challenge / Solution
Debugging / Observability	LangGraph: Clear visualization via graph tracing. AutoGen: Complex causality, distributed logs. CrewAI: Fragmentation across specialized agents.	Requires purpose-built MAES / Eval360 for full observability.
Extensibility / Tooling	LangGraph: Tightly coupled with LangChain. AutoGen: Generic, reusable agents. CrewAI: Customizable roles, third-party integration.	MCP standardizes external tool interfacing.
Human-in-the-Loop	LangGraph: Structured hooks for input/review. AutoGen: Natural conversational flow. CrewAI: Structured human checkpoints.	AutoGen provides the most flexible interactive model.

TABLE VI
FRAMEWORK CONSIDERATIONS ACROSS LANGGRAPH, AUTOGEN, AND CREWAI

Consideration	LangGraph	AutoGen	CrewAI
Debugging / Observability	Clear visualization via graph tracing	Complex causality, distributed logs	Fragmentation across specialized agents

Extensibility / Tooling	Tightly coupled with LangChain structure	Generic, reusable agents	Customizable roles and third-party integration
Human-in-the-Loop	Structured hooks for input / review	Natural part of conversa- tional flow	Defined hu- man check- points

XI. ACKNOWLEDGEMENT

The authors acknowledge the foundational contributions of the developers and research teams behind LangGraph, AutoGen, CrewAI, and the Model Context Protocol specifica- tion, whose open contributions enable advanced comparative architectural studies of this nature.

REFERENCES

[1] IBM, "Crew AI: Enhancing team intelligence through AI collaboration," IBM Think Blog, 2025.
[2] CrewAI Inc., "CrewAI Framework," GitHub Repository, 2025.
[3] Model Context Protocol, "MCP Specification (2025-06-18)," ModelCon- textProtocol.io, 2025.
[4] OpenAI, "Model Context Protocol (MCP) for Agents," OpenAI GitHub Documentation, 2025.
[5] Microsoft Research, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework," Microsoft Research Publi- cations, 2025.
[6] LangChain, "LangGraph Framework," LangChain Documentation, 2025.
[7] O. Husiev, "Multi-Agent Coordination Patterns and Architectures Be- yond the Hype," Medium, 2025.
[8] DataCamp, "CrewAI vs LangGraph vs AutoGen: Which Framework to Choose?," DataCamp Tutorials, 2025.
[9] TrueFoundry, "AutoGen vs LangGraph: A Practical Comparison," True- Foundry Blog, 2025.
[10] Vectara, "MCP: The Control Plane of Agentic AI," Vectara Blog, 2025.
[11] Latenode, "LangGraph vs AutoGen vs CrewAI: Framework Comparison and Architecture Analysis (2025)," Latenode Blog, 2025.
[12] Nitor Infotech, "Agentic AI Frameworks and Their Hidden Complexity,"Nitor Infotech Blog, 2025.
[13] Botpress, "Multi-Agent Evaluation Systems for LLMs," Botpress Blog, 2025.
[14] Arize AI, "AI Agent Frameworks: A Modern Overview," Arize Blog, 2025.
[15] Google Cloud, "What Is a Multi-Agent System?," Google Cloud Dis- cover, 2025.
[16] Cwan Engineering, "Building Multi-Agent Systems with LangGraph," Medium, 2025.
[17] NVIDIA Developer, "An Easy Introduction to LLM Reasoning, AI Agents, and Test-Time Scaling," NVIDIA Blog, 2025.
[18] SparkCo, "LangGraph vs CrewAI vs AutoGen 2025: Deployment Show- down," SparkCo AI Blog, 2025.
[19] Llumo AI, "Multi-Agent System Observability Frameworks for Reliable Debugging," Llumo Blog, 2025.
[20] Galileo AI, "Debugging Multi-Agent AI Systems," Galileo Blog, 2025.
[21] D. Gupta, "Complete Guide to AI Tokens: Optimization and Cost Management," Deepak Gupta Blog, 2025.
[22] M. Charan, "Comprehensive Comparison of AI Agent Frameworks,"