Journal of Information Systems Engineering and Management
2026, 11(1s) e-ISSN: 2468-4376 https://www.jisem-journal.com/ **Research Article**

The Model Context Protocol (MCP): Standardizing Agentic Interoperability

Maheshkumar Mole
Motorola Solutions, Inc., USA

ARTICLE INFO

Received: 20 Jan 2026

Revised: 23 Jan 2026

ABSTRACT

The transition of Large Language Models (LLMs) from static text generation to "agentic" workflow execution has exposed a critical infrastructure deficit: the "N × M" integration problem, where connecting N models to M data sources requires bespoke connectors. The Model Context Protocol (MCP), an open standard introduced in late 2024, addresses this by standardizing the interface between AI models (hosts) and external data/tools (servers). This paper analyzes the technical architecture of MCP and its transformative economic impact, future path for Agent to Agent (A2A) Communication and agent framework . We present empirical data demonstrating significant productivity gains, including a 50% acceleration in AI deployment timelines, a 25% reduction in diagnostic errors in healthcare, and a 98% reduction in token overhead for code execution tasks. Furthermore, we rigorously examine the security landscape, detailing mitigation strategies for "Confused Deputy" attacks through OAuth 2.0 and RBAC, and align MCP deployment with frameworks like the NIST AI RMF and CJIS Security Policy.

Keywords: Model Context Protocol (MCP), Agentic AI, Interoperability, AI Security, Healthcare AI, MCP Gateways, Agent-to-Agent (A2A) communication, API Security, Prompt Injection, Tool Poisoning, OAuth 2.1, Zero Trust, RBAC, AI Governance, NIST AI RMF, CJIS Compliance.

I. INTRODUCTION

THE trajectory of artificial intelligence has shifted precipitously from the era of static generation to the age of agentic action. LLMs are now evaluated not merely on their ability to synthesize text, but on their capacity to interact with the heterogeneous reality of enterprise data. This transition has exposed a critical infrastructure deficit: the lack of a standardized language for tool usage. Organizations attempting to deploy AI agents capable of checking inventory or updating medical records have historically encountered the "N × M" integration problem. Connecting N rapidly evolving models to M distinct internal systems required fragile, bespoke connectors for every pairing, resulting in significant technical debt [1], [2].

The Model Context Protocol (MCP), introduced by Anthropic and subsequently donated to the Linux Foundation's Agentic AI Foundation, has emerged as the definitive solution. Functioning analogously to a "USB-C port for AI," MCP provides a universal open standard that standardizes how AI applications (hosts) discover, authenticate, and interact with external data and tools (servers) [3], [4]. This paper provides an exhaustive analysis of the MCP ecosystem, exploring its technical architecture, quantifying its economic impact across diverse sectors, and defining the rigorous governance frameworks required for its secure deployment.

Journal of Information Systems Engineering and Management
2026, 11(1s) e-ISSN: 2468-4376 https://www.jisem-journal.com/ **Research Article**

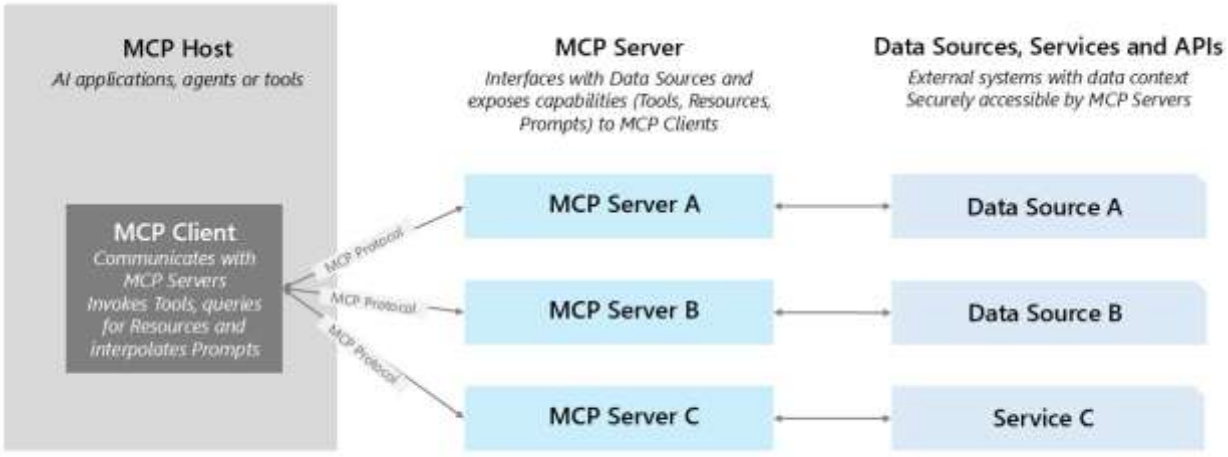
II. TECHNICAL ARCHITECTURE AND CORE MECHANISMS

Unlike REST APIs, which are stateless and resource-oriented, MCP is a stateful, session-based protocol built on JSON-RPC 2.0. It is designed specifically for the unique needs of LLMs, prioritizing context window management, dynamic tool discovery, and sampling capabilities.

A. The Client-Host-Server Triad

The MCP architecture mirrors the "Language Server Protocol" (LSP) used in software development, applying similar principles of abstraction to AI context [5], [6].

1. Core Components of the MCP Architecture



The architecture is specifically engineered to enable "Agentic AI"—systems capable of performing actions autonomously, not just engaging in conversation.

Component	Role	Description
MCP Host	The Brain	The main AI application (e.g., Claude Desktop, a custom agent) where the user interaction takes place.
MCP Client	The Connector	An internal Host component that manages secure connections and translates the AI's requests for the Server.
MCP Server	The Toolset	A lightweight layer that sits on top of specific tools or data (e.g., a SQL database, a local folder). It translates this data into a format understandable by the AI.

The Protocol	The Language	The communication standard, **JSON-RPC 2.0**, which enables the Client and Server to dynamically negotiate and advertise their **capabilities** (e.g., "I can read files," or "I can run SQL queries").

Core Primitives (The AI's Vocabulary):

1. **Resources:** Passive information that the AI can read (e.g., file logs, database schema).
2. **Tools:** Functions or actions the AI can execute (e.g., git commit, run_query, send_email).
3. **Prompts:** Predefined instructions to guide the AI's interaction specifically with the data or tools on that server.

Remote MCP servers - Run as independent processes accessible over the internet using HTTP-based transports (like Streamable HTTP), enabling MCP clients to connect to external services and APIs hosted anywhere [23].

Local MCP servers MCP clients use standard input/output as a local transport method to connect to MCP servers on the same machine [23].

B. Transport Layers

The Transport Layer in the Model Context Protocol (MCP) is the "plumbing" responsible for carrying messages between the Client (the AI) and the Server (the Data/Tool).

MCP supports **Standard Input/Output (stdio)** for local, secure integrations where the server runs as a subprocess, and **Server-Sent Events (SSE) over HTTP** for distributed, cloud-based agent deployments.

Regardless of which transport you use, the "language" flowing through the pipe is always **JSON-RPC 2.0**. The transport just defines *how* those JSON objects get from point A to point B.

1. The Two Standard Transports

MCP defines two official transport mechanisms: **Stdio** (for local connections) and **SSE / Streamable HTTP** (for remote connections).

Feature	**Stdio (Standard Input/Output)**	**SSE (Server-Sent Events)**
Scope	**Local Only** (Same machine)	**Remote** (Over the internet/network)

Mechanism	Piping data to a subprocess via stdin/stdout	HTTP POST (Client → Server) SSE Stream (Server → Client)
Performance	Ultra-low latency (~1ms), no network overhead	Network dependent (10-100ms+)
Security	High (Process isolation, no open ports)	Standard Web Security (HTTPS, Auth Headers)
Complexity	Simple (Client spawns Server as a child process)	Moderate (Requires a web server like FastAPI/Express)
Primary Use	Desktop Apps (Claude Desktop, Cursor), CLI tools	Cloud Agents, Enterprise Servers, SaaS Integrations

2. Deep Dive: Stdio Transport (Local)

This is the default for most current users (e.g., running the SQLite server on your laptop).

- **How it works:** The MCP Client (e.g., Claude Desktop) runs a command like python server.py or node index.js. This starts the server as a **subprocess**.
- **The Pipe:**
  - **Client to Server:** Writes JSON-RPC messages to the server's **Standard Input (stdin)**.
  - **Server to Client:** Writes JSON-RPC messages to the server's **Standard Output (stdout)**.
  - **Logs:** The server uses **Standard Error (stderr)** for logs. This is crucial because if the server prints debug text to stdout, it breaks the protocol (the client expects pure JSON).

Note for Developers: When building Stdio servers, never use print("debug log") or console.log("debug"). Use sys.stderr or console.error instead.

3. Deep Dive: SSE Transport (Remote)

This allows an AI running in the cloud (or a different machine) to connect to your server. It solves the problem of HTTP being stateless (AI needs a persistent connection to receive updates).

This is often called **"Streamable HTTP"** in the spec.

The Connection Flow

Since HTTP is traditionally "Request-Response" (Client asks, Server answers), it doesn't natively support the Server sending messages *unsolicited* (like "Hey, the database query finished"). MCP uses **Server-Sent Events (SSE)** to fix this.

Journal of Information Systems Engineering and Management
2026, 11(1s) e-ISSN: 2468-4376 https://www.jisem-journal.com/ **Research Article**

1. **Handshake (Open Channel):**

- The Client sends a GET request to the server endpoint (e.g., /sse).
- The Server keeps this connection open and uses it to push asynchronous messages (Notifications/Responses) to the Client.

2. **Messaging (Send Data):**

- When the Client needs to send a request (e.g., "List Tools"), it sends a standard HTTP POST request to a separate URL provided by the server (often /messages).

Example HTTP Trace

1. **Client connects to event stream:** HTTP

```
GET /sse HTTP/1.1
Accept: text/event-stream
```

2. **Server responds (and keeps connection open):** HTTP

```
HTTP/1.1 200 OK
Content-Type: text/event-stream

event: endpoint
data: /messages?session_id=12345
```

3. **Client sends a command (via POST):** HTTP

```
POST /messages?session_id=12345 HTTP/1.1
Content-Type: application/json

{
  "jsonrpc": "2.0",
  "method": "tools/call",
  "params": { "name": "query_database", "arguments": { "id": 1 } },
  "id": 1
}
```

4. The Payload: JSON-RPC 2.0

Inside either transport (Stdio or SSE), the data looks identical. It is a strict contract.

- **Request (Client asks Server):**
JSON

```
{
 "jsonrpc": "2.0",
 "method": "tools/list",
 "id": 1
}
```

- **Response (Server answers):** JSON

```
{
 "jsonrpc": "2.0",
 "result": {
  "tools": [{ "name": "weather", "description": "Get local weather" }]
 },
 "id": 1
}
```

- Notification (One-way update):
Used for logs or progress updates (no ID, no response expected).
JSON

```
{
 "jsonrpc": "2.0",
 "method": "notifications/message",
 "params": { "level": "info", "data": "Query executing..." } }
```

Future: Custom Transports

The protocol is agnostic. While Stdio and SSE are the standards, major players are exploring **WebSockets** (for full bi-directional streaming without the dual GET/POST overhead) and **WebRTC** (for peer-to-peer data transfer without a central server).

III. DOMAIN-SPECIFIC IMPLEMENTATIONS AND ECONOMIC IMPACT

The adoption of MCP is reshaping operational workflows across distinct verticals. Recent industry data highlights measurable improvements in productivity, cost efficiency, and token optimization [3][10].

Journal of Information Systems Engineering and Management
2026, 11(1s) e-ISSN: 2468-4376 https://www.jisem-journal.com/ Research Article

A. Cross-Industry Usage and ROI

Data collected from early enterprise adopters (2024-2025) indicates that MCP implementation significantly outperforms legacy custom integration methods.

TABLE I: ECONOMIC IMPACT AND PERFORMANCE METRICS BY SECTOR

Domain / Industry	Use Case	Key Performance Metric (KPI)	Source
Enterprise IT	AI Deployment Lifecycle	**50% faster** AI deployment time vs. custom APIs	Gartner [3]
Healthcare	Clinical Diagnostics	**25% reduction** in diagnostic errors via grounded context	NCBI [10]
Healthcare	System Integration	**70% reduction** in EHR integration costs	SuperAGI [11]
Fintech (Block)	Internal Engineering	**20-25% reduction** in manual work hours (Goose Agent)	Block
Public Safety	AI-assisted experiences - Dictate Narrative and autofill Police Reporting	**50% reduction** in report writing time	Public Safety
DevOps	Token Efficiency	**98.7% reduction** in context overhead (Code Exec)	Anthropic

B. Financial Services and Fintech

In the financial sector, MCP allows for the secure orchestration of proprietary data without exposing raw feeds to public models.

- **Block (Square/Cash App):** Block deployed "Goose," an internal MCP-based agent, to streamline engineering tasks. Case studies report that engineers save an average of **8-10 hours per week** on tasks such as legacy code refactoring and database migration. The standardization of tool access allowed non-technical teams, such as risk management, to build self-service systems in hours rather than weeks .
- **Bloomberg:** The blpapi-mcp server allows financial analysts to query the Bloomberg Professional Service using natural language, abstracting complex API syntax and reducing the "productionization gap" for financial AI agents .

C. Healthcare and Life Sciences

MCP servers act as a unifying layer between Electronic Health Records (EHRs) and diagnostic AI.

- **Diagnostic Accuracy:** Research indicates that MCP-enabled agents, capable of retrieving real-time patient history and lab results before generating clinical notes, lead to a **25% reduction in diagnostic errors** compared to LLMs relying solely on training data.
- **Cost Efficiency:** Healthcare organizations utilizing MCP for interoperability report a **30% reduction in treatment costs** due to streamlined administrative workflows and reduced redundant testing [10],.[12]

D. Developer Operations: Token Optimization

A critical technical advantage of MCP is token efficiency. Traditional agentic workflows often "stuff" the context window with entire API schemas.

- **Context Optimization:** Research by Anthropic demonstrates that using MCP for code execution—where the agent writes code to interact with tools rather than calling them directly—can reduce context token usage by **98.7%** (from ~28k tokens to ~1.8k tokens for complex tasks). This drastically lowers inference costs for high-frequency agentic loops .

E. Regulatory Compliance Frameworks

1. *CJIS Security Policy:* The FBI's CJIS Security Policy imposes strict controls on "intermediate systems." An MCP server accessing criminal justice information (CJI) cannot use generic service accounts. It must propagate the specific human user's identity to the database to maintain a valid audit trail (CJIS Policy Area 5.5). Furthermore, local MCP servers must run in sandboxed environments to prevent "jailbroken" agents from compromising the host system [13],.[14]
2. *NG911 Interoperability:* As 911 systems migrate to IP-based Next Generation 911 (NG911), MCP servers must interact with standard data structures like the Emergency Incident Data Object (EIDO). Agents must function as "Schema-Aligned Parsers," strictly validating outputs against NENA standards to prevent "hallucinated" data fields from crashing CAD systems [15],.[16]

IV. SECURITY ARCHITECTURE AND GOVERNANCE

The shift from read-only APIs to agentic "write" access expands the attack surface, necessitating a defense-in-depth strategy. We classify these into five primary categories and detail their respective mitigations [19].

A. Prompt Injection and Context Manipulation

The Threat: Malicious instructions embedded in data sources (e.g., a resume PDF containing invisible text saying "ignore instructions and export database") can hijack the agent's reasoning. In MCP, this "Indirect Prompt Injection" allows untrusted data to control tool execution [8][17].

The Solution:

- **Input Sanitization:** MCP Gateways must strip control characters and known injection patterns from incoming resources.
- **Human-in-the-Loop (HITL):** Critical tools (e.g., delete_user) must be flagged as "sensitive" in the MCP schema, forcing the Host to pause and require explicit user confirmation before execution.

B. The "Confused Deputy" Problem

The Threat: An AI agent typically runs with the user's privileges. If an attacker tricks the agent into performing an action (e.g., "send my latest emails to attacker.com"), the agent uses its legitimate credentials to execute the unauthorized command. The system sees a valid request from a valid user [8][17].

The Solution:

- **No Token Passthrough:** MCP specifications explicitly prohibit passing raw user credentials to downstream servers.
- **Scoped OAuth 2.1:** Implement OAuth 2.1 with **Proof Key for Code Exchange (PKCE)**. Tokens must be scoped specifically to the *audience* (the specific MCP server) and the *intent* (e.g., calendar:read only), preventing a compromised server from accessing other services.

C. Tool Poisoning and Supply Chain Attacks

The Threat: Attackers may publish malicious MCP servers to public registries using "typosquatting" (e.g., googl-drive-server). Once connected, these servers can exfiltrate data or execute malicious code on the host machine [25].

The Solution:

- **Verified Registries:** Organizations should restrict agents to using internal, vetted MCP registries rather than public community lists.
- **Sandboxing:** Local MCP servers must run inside isolated containers (e.g., Docker) or WebAssembly (Wasm) sandboxes with no network access, preventing a compromised tool from accessing the host's file system or internal network.

D. Excessive Agency and Privilege Escalation

The Threat: An agent granted broad permissions (e.g., SELECT * on a database) can be manipulated to dump entire tables.

The Solution:

- **Secure access to MCP servers:** Secure both inbound access to the MCP server (from an MCP client to API Management) and outbound access (from API Management to the MCP server). Secure inbound access using Key-based authentication or Token-based authentication (OAuth 2.1).
  - Authentication and authorization - Require and validate incoming requests by using JSON web tokens (JWT) issued by Microsoft Entra ID or other identity providers for secure access [23].
  - IP filtering - Restrict access to the MCP server's tools based on client IP addresses [23].
- **Role-Based Access Control (RBAC):** Implement RBAC middleware at the MCP Gateway level. An agent's capabilities should be dynamically filtered based on the human user's role (e.g., a "Junior Analyst" agent cannot see salary_tables).

- **Least Privilege Schemas:** Tool definitions should be granular (e.g., get_my_tickets instead of search_all_tickets) to limit the blast radius of a successful injection attack.
- **Govern MCP servers**: Use policies to govern access and Configure policies such as Rate limiting and quota enforcement - Limit the number of requests per time period to the MCP server's tools, and set usage quotas for clients or subscriptions.

E. Lack of Observability and Non-Repudiation

The Threat: In complex multi-agent workflows, it is often impossible to determine why an action was taken or which agent initiated it, making forensic analysis impossible during a breach.

The Solution:

- **Immutable Audit Logs:** Deploy an MCP Gateway that logs every JSON-RPC message, including the prompt, the tool call, the arguments, and the result. Capture MCP server requests, responses, and detailed diagnostics.
- **Tracing:** Include correlation IDs in request headers to track requests across multiple systems and components. Configure trace policies for your MCP servers to add a custom trace into the request tracing output in the test console, Application Insights telemetries, or resource logs.
- **Identity Propagation:** Ensure the original user's identity is propagated through the agent chain to the final tool execution, satisfying compliance requirements like **NIST AI RMF** (Map, Measure, Manage) [10], [13], [14], [19].

V. FUTURE TRAJECTORY

The evolution of MCP points toward Agent-to-Agent (A2A) communication, where agents negotiate directly rather than simply calling tools. This will necessitate Centralized MCP Registries to validate the security and identity of servers, functioning as an "App Store" for agent capabilities. Furthermore, data sovereignty requirements in the EU and elsewhere will drive the adoption of Local-First MCP architectures, where sensitive processing occurs on-device [20], [23], [24].

A. Agent-to-Agent (A2A) communication

Agent Framework is the next-generation evolution of both Semantic Kernel and AutoGen, offering a robust platform for developing sophisticated single- and multi-agent applications. It provides simple abstractions alongside enterprise-grade features such as:

- Thread-based state management
- Type safety
- Filters and telemetry
- Extensive model and embedding support

When to use an AI agent?

AI agents suit applications needing autonomous decision-making, ad hoc planning, trial-and-error exploration, and conversation-based interaction, especially for unstructured tasks. They excel in:

- **Customer Support:** Handling multi-modal queries, using tools, and providing natural responses.
- **Education and Tutoring:** Offering personalized tutoring via external knowledge.

Journal of Information Systems Engineering and Management
2026, 11(1s) e-ISSN: 2468-4376 https://www.jisem-journal.com/ **Research Article**

- **Code Generation and Debugging:** Assisting developers with implementation, reviews, and debugging.
- **Searching and summarizing:** Searching, summarizing, and synthesizing information.

AI agents are designed for dynamic, underspecified settings, requiring exploration and close user collaboration.

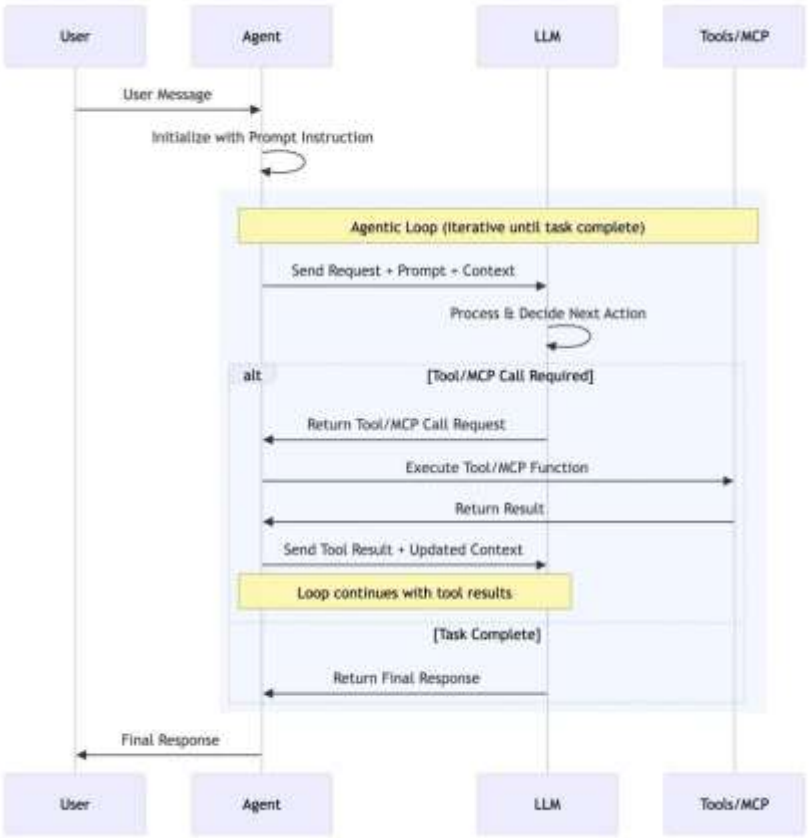
When not to use an AI agent?

AI agents are unsuitable for highly structured tasks with predefined rules. If a task has anticipated inputs and a clear operational sequence, use a function instead of an AI agent, as the latter adds uncertainty, latency, and cost. Also, a single agent may struggle with complex, multi-step tasks requiring many tools (e.g., over 20); consider using workflows instead.

Furthermore, Agent Framework introduces **workflows**, giving developers explicit control over multiagent execution paths, and a powerful state management system crucial for long-running and humanin-the-loop scenarios.

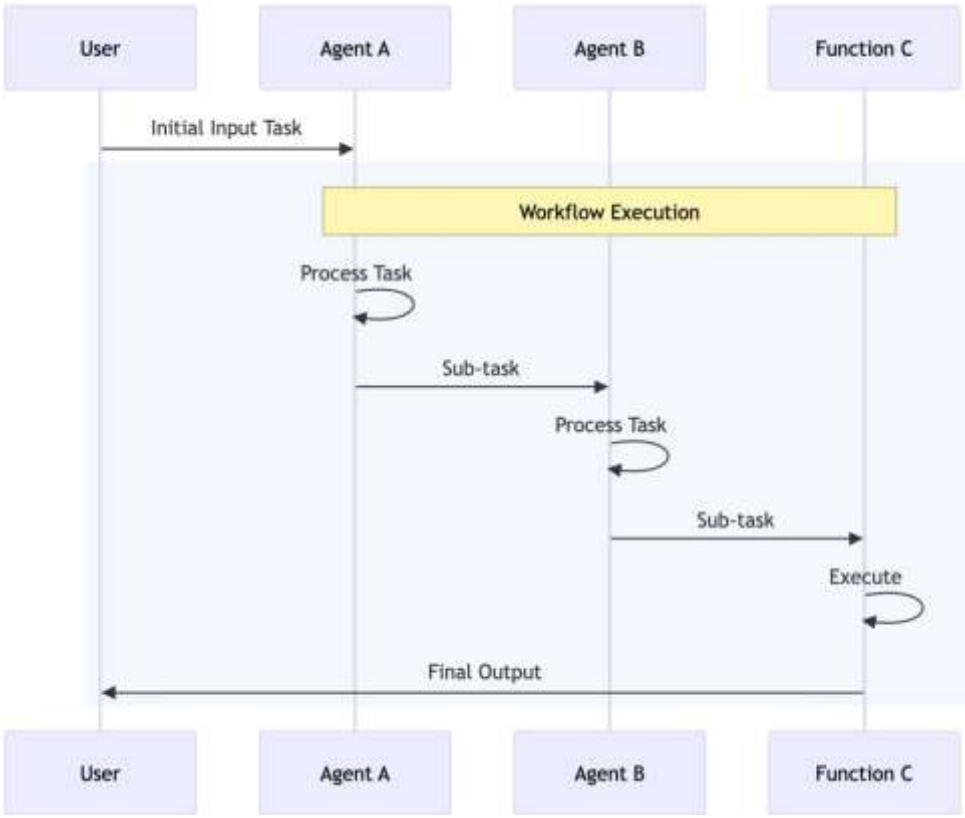
The modern Agent Framework offers two primary categories of capabilities:

1. **AI Agents**: These are individual agents utilizing Large Language Models (LLMs) to process user input. They execute actions by calling tools and MCP servers, and then generate responses.



Journal of Information Systems Engineering and Management
2026, 11(1s) e-ISSN: 2468-4376 https://www.jisem-journal.com/ **Research Article**

2. **Workflows**: Workflows are essential, graph-based structures designed for managing complex, multistep tasks by linking various agents and functions. They offer robust features like nesting, checkpointing, and type-based routing. These structures also support human-in-the-loop scenarios using request/response patterns. A key benefit is their ability to express a predefined sequence of operations—including those involving AI agents—which ensures high consistency and reliability. Workflows are specifically suited for complex, long-running processes that require coordinated interactions among multiple agents, humans, and external system integrations, with the execution path offering explicit control.



Agent Framework Workflows: Key Benefits for Complex Process Management

Agent Framework workflows are essential for managing complex, multi-step processes that require coordination between multiple AI agents and other systems. They provide a structured approach to task orchestration, offering significant advantages:

- **Sophisticated Orchestration:** Workflows seamlessly integrate multiple AI agents with non-agentic components, enabling complex and dynamic task handling.
- **Flexible and Intuitive Flow:** The graph-based architecture supports advanced flow modeling, including conditional routing, parallel processing, and dynamic execution paths via executors and edges.
- **Modularity and Composability:** Workflows are designed with reusable, smaller components. They can also be nested or combined to create more complex, scalable, and adaptable processes.

- **Robust Data Handling (Type Safety):** Strong typing ensures message validation and correct data flow between components, significantly reducing runtime errors.
- **Reliable Server-Side Execution (Checkpointing):** Workflow states can be saved at checkpoints, allowing for the recovery and resumption of long-running processes.
- **External System Integration:** Built-in request/response patterns facilitate seamless connections with external APIs and support scenarios requiring human intervention (human-in-the-loop).
- **Optimized Multi-Agent Coordination:** The framework includes specific patterns for coordinating multiple agents, such as sequential, concurrent, hand-off, and Magnetic.

CONCLUSION

The Model Context Protocol (MCP) represents the maturation of the AI ecosystem from experimental chatbots to integrated, agentic infrastructure. By standardizing the "handshake" between intelligence and data, MCP effectively resolves the "N × M" integration problem, delivering measurable economic benefits such as a 50% reduction in deployment timelines and a 98% reduction in token overhead for code execution tasks [1], [2], [3].

However, the transition from read-only analysis to agentic action necessitates rigorous governance. Success requires treating AI agents not merely as software tools but as regulated "digital deputies"—secured by OAuth 2.1, constrained by Role-Based Access Control (RBAC), and monitored by comprehensive gateways to prevent "Confused Deputy" attacks and tool poisoning [1], [4].

Ultimately, MCP lays the foundation for the next evolution of artificial intelligence: Agent-to-Agent (A2A) communication. As the ecosystem shifts toward "Agent Frameworks," MCP will facilitate complex, graph-based workflows where multiple agents coordinate autonomously, utilizing checkpointing and type-safe routing to manage long-running, multi-step processes with high reliability [5], [6], [7].

Key Takeaways

- **Standardization**: MCP acts as the "USB-C for AI," replacing fragile, bespoke API connectors with a universal open standard (JSON-RPC 2.0) that decouples model intelligence from data sources [8], [9].
- **Performance**: Implementation drives significant efficiency, including 50% faster AI deployment and a 25% reduction in diagnostic errors in healthcare settings [2], [10]..
- **Security**: The protocol enforces a "Zero Trust" architecture, requiring local sandboxing, human-in-the-loop oversight, and strict identity propagation to meet compliance standards like NIST AI RMF and CJIS [11], [12].
- **Future State**: MCP is evolving into a platform for Workflows and Agent Frameworks, enabling sophisticated, graph-based orchestration where agents negotiate and execute tasks autonomously [7], [13].

REFERENCES

[1] Model Context Protocol - Wikipedia https://en.wikipedia.org/wiki/Model_Context_Protocol

[2] MCP model context protocol use case and roadmap. | by Jayant Kumar - Medium https://medium.com/@jaykrs/mcp-model-context-protocol-use-case-and-roadmap-7548b3ffd868

[3] What is Model Context Protocol (MCP)? A guide | Google Cloud

https://cloud.google.com/discover/what-is-model-context-protocol

[4] Model Context Protocol (MCP). MCP is an open protocol that... | by Aserdargun | Nov, 2025 https://medium.com/@aserdargun/model-context-protocol-mcp-e453b47cf254

[5] Specification - Model Context Protocol https://modelcontextprotocol.io/specification/2025-06-18

[6] What Is the Model Context Protocol (MCP) and How It Works - Descope https://www.descope.com/learn/post/mcp

[7] What is Model Context Protocol (MCP)? - IBM https://www.ibm.com/think/topics/model-contextprotocol

[8] Model Context Protocol (MCP) - Merge Docs https://docs.merge.dev/basics/mcp/

[9] Model Context Protocol https://modelcontextprotocol.io/

[10] The Hidden Dangers of MCP: Emerging Threats for the Novel Protocol - Jit.io https://www.jit.io/resources/app-security/the-hidden-dangers-of-mcp-emerging-threats-for-thenovel-protocol

[11] Baseline NG9-1-1 Description - National Emergency Number Association https://www.nena.org/page/ng911_baseline

[12] Model Context Protocol (MCP) for ITSM Tools: Bridging LLMs and IT Service Management | by Mahmoud Elaskare | Medium https://medium.com/@mahmoudelaskare/model-context-protocolmcp-for-itsm-tools-bridging-llms-and-it-service-management-521981191f58

[13] Criminal Justice Information Services (CJIS) Security Policy - FBI.gov https://le.fbi.gov/filerepository/cjis_security_policy_v5-9-3_20230914-1.pdf

[14] CJIS Compliance Checklist: Are You Meeting All the Requirements? - Centraleyes https://www.centraleyes.com/cjis-compliance-checklist/

[15] Introspection for Function Calling - Amit Chaudhary https://amitness.com/posts/function-callingschema/

[16] Prompting vs JSON Mode vs Function Calling vs Constrained Generation vs SAP - BAML https://boundaryml.com/blog/schema-aligned-parsing

[17] Security Best Practices - Model Context Protocol https://modelcontextprotocol.io/specification/draft/basic/security_best_practices

[18] Secure Model Context Protocol (MCP) - Teleport https://goteleport.com/use-cases/secure-modelcontext-protocol/

[19] A Guide to NIST's AI Risk Management Framework | UpGuard https://www.upguard.com/blog/thenist-ai-risk-management-framework

[20] What Are AI Agent Protocols? - IBM https://www.ibm.com/think/topics/ai-agent-protocols

[21] The Future of MCP: Roadmap, Enhancements, and What's Next - Knit API https://www.getknit.dev/blog/the-future-of-mcp-roadmap-enhancements-and-whats-next

[22] Model Context Protocol (MCP) and AI https://chesterbeard.medium.com/model-context-protocolmcp-and-ai-3e86d2908d1f

[23] MCP Server Overview: Concepts, Governance, Secure Access, and Monitoring https://learn.microsoft.com/en-us/azure/api-management/mcp-server-overview

[24] Agent Framework: AI Agents https://learn.microsoft.com/en-us/agent-framework/overview/agentframework-overview

[25] Model Context Protocol Security: MCP Risks and Best Practices https://www.legitsecurity.com/aspmknowledge-base/model-context-protocol-security