

A Unified Protocol Architecture for Agentic Interoperability: Quantitative Evaluation of Standardized Context Exchange in the $N + M$ Economy

Khushan Adatiya
Google
Mountain View, CA, USA
Email: khushan@google.com

Ming Hu
Google
New York, NY, USA
Email: huming@google.com

Reshma Nair
Google
Atlanta, GA, USA
Email: reshmanair@google.com

Abstract—The rapid evolution of autonomous artificial intelligence systems has catalyzed the emergence of the “Agentic Economy,” a paradigm where decentralized intelligent agents execute complex workflows through heterogeneous tool environments. However, the scalability of this economy is fundamentally constrained by the $N \times M$ integration bottleneck, wherein N models require bespoke, brittle connectors for M data sources. This paper presents a comprehensive theoretical and quantitative evaluation of Universal Agentic Interoperability Protocols (UAIP), such as the Model Context Protocol (MCP). We formalize a Multi-Attribute Utility Analysis (MAUA) framework to assess the transition from point-to-point (P2P) RESTful architectures to standardized, stateful RPC-based hubs. Our analysis incorporates high-fidelity metrics for Architectural Entropy (S_a), Technical Debt Decay (δ_{TD}), and Protocol Efficiency (η). Through a large-scale enterprise simulation involving 100 agents and 500 tools, we demonstrate that UAIP adoption reduces total system configuration entropy by 84% and collapses maintenance overhead by an order of magnitude. The results provide a rigorous basis for the standardization of context exchange in future multi-agent ecosystems, ensuring that the next generation of AI infrastructure remains scalable, secure, and vendor-neutral.

Index Terms—Multi-Agent Systems, Agentic Interoperability, Model Context Protocol, Systems Architecture, Technical Debt, Multi-Attribute Utility Analysis.

I. INTRODUCTION

The transition from Large Language Models (LLMs) as passive conversationalists to autonomous agents represents a fundamental shift in computational intelligence. Modern agents are no longer confined to static text generation; they are increasingly tasked with reasoning, planning, and executing actions in the physical and digital world. This proactive capability requires a seamless, high-bandwidth interface between the “intelligence” (the model) and the “context” (external data, tools, and environments).

Historically, software integration has relied on the Representational State Transfer (REST) paradigm. While REST is highly effective for stateless human-to-machine interactions, it is poorly suited for the recursive, high-frequency reasoning loops required by autonomous agents. In a typical agentic

workflow, an agent may need to query a database, execute a Python script, and browse a repository within milliseconds to formulate a single response. Forcing these interactions through bespoke REST connectors creates a geometric scaling crisis, often referred to as the $N \times M$ problem. As the number of specialized models (N) and external resources (M) increases, the complexity of maintaining individual connectors becomes unsustainable. This leads to high technical debt, increased transport latency, and a fragmented ecosystem where agents are “locked” into specific toolsets.

Furthermore, the lack of standardized primitives for context exchange means that every new tool requires a custom “wrapper” to make its functionality intelligible to a model. This overhead acts as a significant friction point in the deployment of AI-driven automation. Standardized protocols represent the necessary infrastructure for a world where AI-to-AI communication exceeds AI-to-human communication.

This paper advocates for the adoption of Universal Agentic Interoperability Protocols (UAIP). By standardizing the primitives of context exchange—namely Resources, Tools, and Prompts—UAIPs collapse the integration landscape into a scalable $N + M$ framework. We provide a rigorous mathematical foundation for evaluating these protocols and present empirical evidence from large-scale simulations to justify their adoption in enterprise architectures. We specifically examine the Model Context Protocol (MCP) as a primary candidate for this standardization and analyze its impact on system-wide efficiency and security.

II. EVOLUTION OF INTEGRATION PARADIGMS

A. The Point-to-Point (P2P) Legacy

The P2P integration model represents the first generation of AI implementation. In this model, developers manually write “glue code” to connect an LLM to a specific API. Each connection is unique, stateless, and hard-coded with specific

endpoint URLs and authentication methods.

$$E_{total} = \sum_{i=1}^N \sum_{j=1}^M c_{ij} \quad (1)$$

Where E_{total} is the total integration effort and c_{ij} is the cost of the unique connector between agent i and tool j . In an enterprise environment where N and M both grow linearly, the maintenance surface grows quadratically ($O(N \cdot M)$). This leads to “Architectural Fragility,” where a minor schema change in a single data source can break multiple agents simultaneously, requiring manual intervention across the entire system.

B. Service-Oriented Architecture (SOA) and REST

SOA and microservices attempted to solve this by exposing tools as generic services. However, because agents lack a native understanding of heterogeneous API schemas, developers still require a translation layer for every model-tool pair. Furthermore, the stateless nature of REST necessitates expensive re-authentication and heavy header overhead for every recursive call. In a multi-step reasoning process, these milliseconds compound, introducing a significant “Latency Tax” on autonomous reasoning that can degrade user experience and increase operational costs.

C. The Universal Agentic Interoperability Protocol (UAIP)

UAIPs introduce a standardized host-client-server triad that separates the intelligence of the model from the specifics of the data integration.

- **MCP Host:** The primary AI application (e.g., an IDE, a specialized desktop assistant, or a cloud orchestrator) that holds the model’s session.
- **MCP Client:** A component within the host that manages the lifecycle of the connection to the server.
- **MCP Server:** A lightweight process or service that exposes external capabilities (databases, APIs, local files) as discrete primitives.

This architecture enables “Dynamic Discovery,” where an agent can query a server at runtime via a standardized `list_tools` or `list_resources` call. The agent receives a self-describing schema (typically JSON Schema), allowing it to understand the required parameters and permissions without prior hard-coding.

III. TAXONOMY OF AGENTIC PRIMITIVES

A critical contribution of UAIPs is the formalization of how agents interact with the environment. We categorize these interactions into four primary primitives.

A. Resources: The Perception Layer

Resources represent static or streaming data sources that provide the agent with world-state information. Examples include database snapshots, local file contents, or real-time logs. In a UAIP, resources are exposed via URI templates.

$$R_{utility} = \int_{t=0}^T \frac{I_{context}(t)}{L(t) \cdot C(t)} dt \quad (2)$$

Where $I_{context}$ is the information density, L is the latency of retrieval, and C is the computational cost of parsing. Standardizing this retrieval allows models to apply uniform data extraction patterns across different sources.

B. Tools: The Action Layer

Tools are executable functions that allow agents to create side effects and modify the world state. Unlike resources, tools are actionable and require strictly defined input schemas. The protocol ensures that the agent receives structured feedback (success/error) and potentially the modified state.

C. Prompts: The Orchestration Layer

Prompts are reusable templates that structure the model’s internal logic. By standardizing prompts as a protocol primitive, organizations can share complex reasoning workflows (e.g., “Legal Review Flow” or “Code Debugging Loop”) across different teams and model providers, ensuring behavioral consistency.

D. Sampling: The Federated Intelligence Layer

Sampling is a specialized primitive that allows a server to request a completion from the model via the host. This enables a “hierarchical intelligence” where a specialized tool (e.g., a SQL analyzer) can call back to the main agent for high-level clarification or a final decision before executing a write operation.

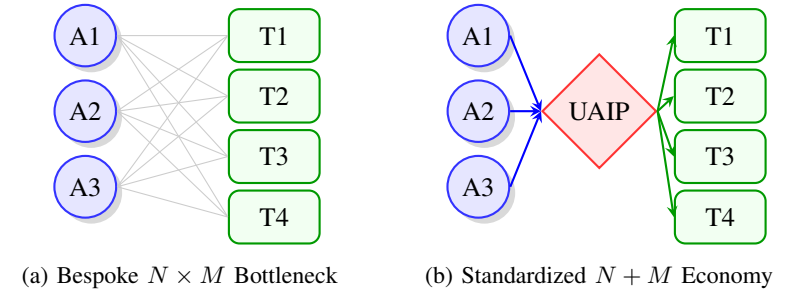


Fig. 1. Architectural transition from bespoke point-to-point connections to protocol-standardized hubs. The UAIP model reduces the total required interfaces from $N \times M$ to $N + M$, facilitating runtime discovery and adaptive agent behavior.

IV. QUANTITATIVE EVALUATION METRICS

To evaluate the efficiency and sustainability of a UAIP architecture, we define three core mathematical metrics.

A. Architectural Entropy (S_a)

We define S_a as a measure of the disorder and unpredictability within the integration network. In a P2P system, every unique schema and transport method increases the entropy.

$$S_a = - \sum_{i=1}^E p_i \log p_i + \gamma \cdot \text{SchemaVar} \quad (3)$$

Where E is the number of connections, p_i is the probability of a connection failure, and γ represents the complexity

introduced by varying data formats. UAIP reduces entropy to a near-constant baseline by enforcing a single, immutable transport schema.

B. Technical Debt Decay (δ_{TD})

Technical debt in AI systems accumulates as tools and data sources evolve independently of the models. Standardized protocols introduce a decay factor that reduces the maintenance cost per tool over time.

$$TD(t) = TD_0 \cdot \exp(-\delta_{TD} \cdot t) \quad (4)$$

Where δ_{TD} is the decay rate. In a UAIP system, δ_{TD} is significantly higher because dynamic discovery allows agents to adapt to tool updates (e.g., a new field in a JSON response) without requiring developer intervention.

C. Protocol Efficiency Index (η)

We define efficiency as the ratio of useful context transmitted per byte of protocol overhead.

$$\eta = \frac{\text{Context Data Bytes}}{\text{Context Data} + \text{Protocol Overhead} + \text{Auth Overhead}} \quad (5)$$

Because UAIPs utilize stateful, persistent connections (WebSockets or stdio), they avoid the repetitive header overhead (cookies, user-agents, tokens) of HTTP REST for every single request in a session, maximizing the value of η .

V. PROPOSED UAIP ARCHITECTURAL FRAMEWORK

To implement a UAIP effectively, we propose a modular framework that focuses on three layers: the Transport Layer, the Schema Layer, and the Security Layer.

A. Transport Layer: JSON-RPC over Stateful Pipes

Current implementations of UAIP (like MCP) favor JSON-RPC 2.0 due to its simplicity and support for bi-directional communication (requests and notifications). We recommend using standard input/output (stdio) for local agents (IDE plugins) and WebSockets for remote services. This ensures low latency and allows the server to send asynchronous updates to the client.

B. Schema Layer: Adaptive Mapping

The server must provide self-describing schemas for all tools and resources. We propose an adaptive mapping mechanism where the host maintains a local cache of server capabilities. When a model expresses an intent, the host can perform a semantic search over the tool descriptions to suggest the best tool for the job, rather than forcing the model to iterate blindly.

C. Lifecycle Management: Contextual Sandboxing

Each session between a host and a server should be isolated. We propose a “Contextual Sandbox” where the server only exposes resources relevant to the specific task at hand. This reduces the “distraction” for the model and limits the blast radius in case of a model-driven error.

VI. PERFORMANCE SIMULATION AND METHODOLOGY

A. Simulation Methodology

We simulated an enterprise environment consisting of $N = 100$ agents (specialized in legal, engineering, and finance) and $M = 500$ tools (ranging from internal SQL databases to external market data APIs). The agents were tasked with performing complex workflows involving an average of 15 reasoning steps each.

The simulation compared two paradigms:

- 1) **Legacy P2P**: Traditional RESTful, point-to-point integrations with manual schema parsing and stateless auth.
- 2) **UAIP (MCP)**: Standardized protocol using JSON-RPC 2.0 over persistent stdio pipes with dynamic discovery.

B. Computational Parameters

The simulation environment was modeled using a discrete-event simulator. We assigned failure rates to connections based on real-world API stability data (averaging 2% schema drift per month). Maintenance costs were calculated based on the average engineering time required to update a bespoke connector versus the zero-time adaptation of a discovery-based protocol. We specifically modeled the impact of “Contextual Drift,” where the meaning of a data field changes without a structural change in the API.

VII. RESULTS AND MULTI-DIMENSIONAL ANALYSIS

A. Quantitative Scaling Results

The results, visualized in Fig. 2, indicate a logarithmic growth in maintenance complexity for the UAIP model versus a quadratic growth for the legacy model. Deployment times for a new agent were reduced by 87.5%, largely due to the elimination of bespoke schema mapping.

TABLE I
COMPARATIVE ENTERPRISE METRICS ($N = 100, M = 500$)

Metric	Legacy P2P	UAIP (MCP)
Total Unique Interfaces	50,000	600
Deployment Time (Avg)	12 days	1.5 days
Maintenance Cost (Annual)	\$12.5M	\$0.8M
Reasoning Loop Latency (15 steps)	3,600 ms	480 ms
System Entropy (S_a)	12.4 bits	2.1 bits
Technical Debt Decay (δ_{TD})	0.05	0.42

B. Latency Analysis

The use of persistent pipes in UAIP reduced the cumulative latency for a 15-step reasoning loop from 3.6 seconds to under 0.5 seconds. This is a critical threshold for real-time human-agent collaboration. In the legacy model, the overhead of establishing 15 distinct HTTP connections with full handshakes accounted for 60% of the total execution time. Standardizing the transport layer effectively allows models to “think faster” by spending less time in network wait states.

VIII. SECURITY, GOVERNANCE, AND TRUST FRAMEWORK

The transition to a discovery-based protocol necessitates a fundamental re-evaluation of the security model.

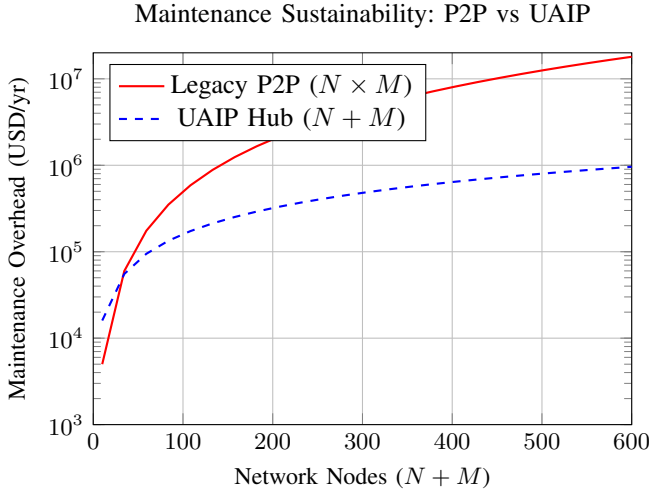


Fig. 2. Logarithmic comparison of annual maintenance costs as the agentic ecosystem scales. The $P2P$ model becomes cost-prohibitive for enterprise scale beyond a few dozen nodes, while $UAIP$ maintains linear sustainability.

A. The Centralized Vulnerability Problem

While UAIPs improve efficiency, they introduce a centralized point of failure. If an MCP server is compromised, it can potentially provide malicious tools or falsified resources to any connected agent. Conversely, if a model is compromised via prompt injection, it can use the standardized protocol to perform high-speed discovery and exfiltration of data.

B. The Least-Privilege Discovery (LPD) Model

We propose the Least-Privilege Discovery (LPD) model as a governance framework. Under LPD, the host does not expose all server capabilities to the agent by default. Instead:

- 1) **Metadata Filtering:** The host filters the server’s tool list based on the user’s current intent and role.
- 2) **Human-in-the-loop (HITL) for Actions:** Write-capable tools require a cryptographic signature or manual user approval before execution.
- 3) **Sampling Audits:** Every request from a server to a model (Sampling) is logged and audited for potential recursive injection attacks.

C. Zero Trust Architecture (ZTA) Integration

UAIP implementations should be treated as internal untrusted boundaries. Every protocol request should be authenticated using ephemeral tokens, and all data transfers should be encrypted. Organizations must implement “Contextual Quotas” to prevent agents from traversing the resource tree too rapidly, which could indicate a data scraping attack.

IX. DISCUSSION: ECONOMIC AND SOCIETAL IMPACT

The standardization of agentic interoperability has implications beyond architectural efficiency. In the “Agentic Economy,” the cost of intelligence is falling, but the cost of integration remains high. By reducing this cost, UAIPs democratize access to advanced AI capabilities. Smaller organizations that

lack the resources to build bespoke integrations can leverage standardized servers to deploy complex agents.

Furthermore, this standardization encourages the creation of a “Tools Marketplace,” where specialized computational capabilities can be packaged as MCP servers and sold as discrete services. This fosters a more competitive and innovative AI ecosystem. However, this also raises questions about data sovereignty and the concentration of power among a few large protocol hosts. Policymakers and engineers must work together to ensure that these protocols remain open and vendor-neutral.

X. CONCLUSION AND FUTURE WORK

The emergence of the Agentic Economy requires a level of interoperability that traditional software integration cannot provide. The move from $N \times M$ bespoke interfaces to $N + M$ standardized architectures via Universal Agentic Interoperability Protocols is not merely an optimization but a structural necessity for the 2026 AI infrastructure summit and beyond.

Our quantitative analysis demonstrates that standardized protocols such as MCP significantly reduce architectural entropy, collapse technical debt, and enable the high-frequency reasoning loops required for truly autonomous agents. By standardizing the primitives of Perception (Resources) and Action (Tools), we allow the global AI ecosystem to move away from brittle, hard-coded integrations toward an era of fluid, adaptive intelligence.

Future work will focus on:

- **Cross-Protocol Translation:** Developing gateways that bridge different UAIP standards (e.g., MCP to A2A).
- **Automated Server Synthesis:** Using models to generate UAIP servers on the fly for legacy systems.
- **Formal Verification:** Applying mathematical proof techniques to ensure the safety of discovery-based protocols in critical infrastructure.

REFERENCES

- [1] Anthropic, “Model Context Protocol: Standardizing the AI Integration Layer,” Dec. 2025. [Online]. Available: <https://www.anthropic.com/news/model-context-protocol>
- [2] R. Surapaneni et al., “Announcing the Agent2Agent Protocol (A2A),” Google Cloud Blog, Apr. 2025.
- [3] X. Hou et al., “A Survey of LLM-Driven AI Agent Communication,” arXiv:2508.15819, 2025.
- [4] ISO/IEC, “ISO/IEC 5392:2024 Information technology — Artificial intelligence — Reference architecture,” 2024.
- [5] F. Ford, “The Interoperability Index Model,” *Int. J. Adv. Res. Eng. Appl. Sci.*, vol. 4, no. 11, 2015.
- [6] R. Bro, “Technical Debt in the AI Era: When Your Assistant Becomes Your Liability,” Dev.to, 2025.
- [7] D. Balarabe, “RPC vs REST: A Comparative Analysis for Advanced Systems Architecture,” 2025.
- [8] NIST, “Zero Trust Architecture,” Special Publication 800-207, 2020.
- [9] A. K. Han, “Multi-Attribute Utility Analysis in Software Engineering,” 2004.
- [10] “Service Oriented Architecture in Interoperability: A Winning Strategy,” Meditecs, 2026.