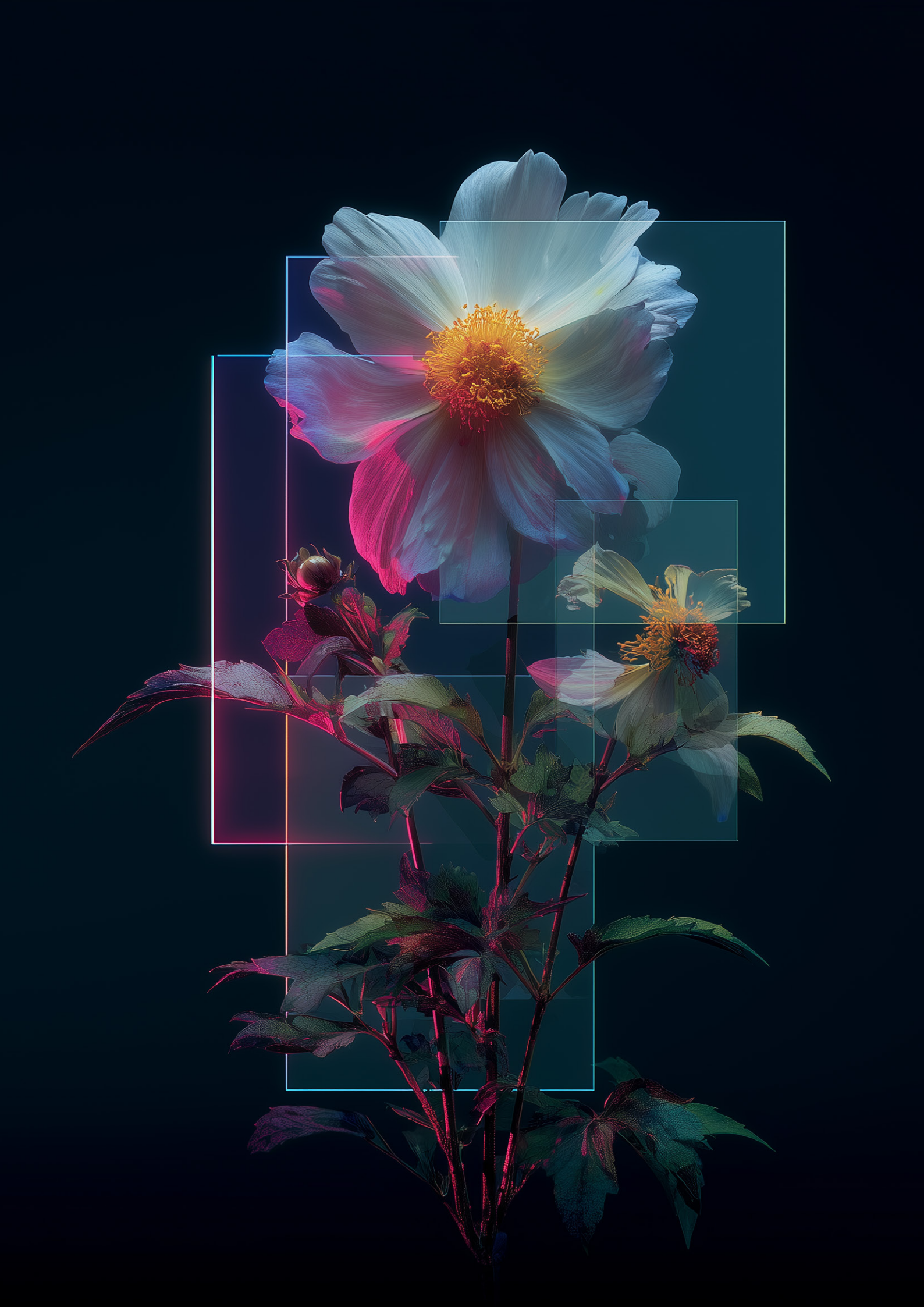
REPLY

WHITEPAPER

Integrating Multi-Agent Systems

The evolution of enterprise architecture: scaling AI through the Model Context Protocol and Agent2Agent.



1 Introduction

The Rise of Multi-Agent Systems (MAS)

The landscape of artificial intelligence (AI) applied to the enterprise world is undergoing a radical change: for a long time, the dominant approach was based on specialised, often monolithic, algorithms and models. These were systems designed as unique entities capable of providing predictive responses based on static and rigidly structured data. These systems, despite their sophistication, were considered sufficient in relatively linear business scenarios, predictable processes, and contexts of limited complexity.

The explosion of Generative AI, the growth of connections between business functions (one need only think, for example, of agile working methods), and the exponential increase in data available in real-time have rapidly highlighted the intrinsic limits of this approach. Operational rigidity and the inability to manage problems requiring decomposition and distributed resolution into sub-tasks have necessitated new and more powerful design paradigms.
In this context, the experience of Reply professionals has led to multi-agent systems being considered as an architecture that was first studied, then tested, and is now increasingly used in production environments. The focus of architects today extends beyond the capabilities of any single AI model, concentrating on the collective abilities of autonomous agents to generate complex, coordinated, and adaptive behaviours, often enhanced by the growing versatility of large language models (LLMs), large action models, computer vision, and other evolutions of generative AI.

The vision of multi-agent systems is founded on the concept of distributed intelligence: each agent acts with decisional autonomy, plans short- and long-term strategies, and interacts proactively with other agents, as well as with external systems, be they digital or physical.
A multi-agent system conceived in this way is more than just an assembly of independent entities: it is an integrated ecosystem in which cooperation, negotiation, communication, and collective learning make it possible to tackle complex and dynamic problems that would be impossible to solve using traditional approaches.

The design of the architecture in the multi-agent system field thus becomes a choice with strategic value, offering the possibility of designing ecosystems composed of a few specialised agents for specific business processes, but also of scaling up to reasoning on architectures based on hundreds of interoperating agents, supporting systemic and multifactorial challenges. Indeed, Reply's experience shows that these architectural decisions are never purely technical; they are profoundly influenced by the nature of the application domain, the complexity of the processes to be supported, and the objectives of scalability, resilience, and specialisation.

Distributed models are proving increasingly effective because they tend to mirror and then improve the processes and organisational structures, collaborative dynamics, and decision-making flows of human societies, thus being intrinsically better suited to managing scenarios that are, by their nature, non-linear and interconnected. Rather than imitating and enhancing the contribution and individual intelligence of each company employee, MAS aim to recreate a collective intelligence on a corporate scale, capable of generating value through the coordination and cooperation of a multitude of autonomous agents.

Integration as the Key to MAS Success

The true power of a multi-agent system lies in the quality of its integrations. Even the most advanced individual agents, endowed with specialised capabilities and sophisticated decisional autonomy, lose much of their value if they are unable to communicate and coordinate effectively, both with each other and with business systems. Without a robust interaction infrastructure, what emerges is a disjointed collection of isolated entities, incapable of generating the collective intelligence that is the promise of MAS.

In several Reply projects, the daily challenges go beyond developing high-performing single agents, extending to the design and implementation of communication and cooperation ecosystems that are solid, flexible, and scalable, capable of guaranteeing operational coherence, adaptability, and resilience in the face of the unexpected. In this context, the focus on integration shifts from defining protocols for data exchange to orchestration with consolidated standards and the construction of complex interaction patterns capable of responding to dynamic business needs.

Only through orchestration does it become possible to manage complex dialogues and resolve conflicts in a coherent and unambiguous manner. The balance between rigidity and flexibility thus represents one of the most delicate design challenges in artificial intelligence projects. Excessively restrictive architectures can stifle the creativity and adaptability of agents when faced with unexpected events, making the system fragile. On the other hand, structures that are too permissive risk generating chaos, inconsistencies, misunderstandings, and potentially catastrophic failures due to the exponential impact of network systems.

This whitepaper attempts to summarise some of the evidence from the various Reply teams engaged in the design, implementation, and integration of multi-agent systems for clients operating in different countries and in the many industries that are rapidly adopting them. It does not claim to be exhaustive or definitive in its conclusions, given the effervescence and speed at which AI is evolving; rather, it aims to be an element to support C-level executives in understanding how the integration of multi-agent systems has strong business implications, beyond the many technological aspects that need monitoring.

2 A Comparison of Protocols

Integration and Communication Protocols as Connective Tissue

In the context of multi-agent systems, communication protocols act as the connective tissue that allows any set of distributed agents to operate as a cohesive, purpose-driven entity. They represent the set of formal rules, technical specifications, and shared standards that unambiguously define the methods by which agents can find each other on a network (so-called “discovery”), communicate their capabilities, take on tasks, and exchange information flows in a way that is at once secure, reliable, and efficient.

It is fundamental to understand that these protocols transcend the function of simple data transmission channels: integration operates on multiple interconnected levels, each indispensable to the system’s success.

Syntax defines the common grammar, the structure, and the format of the exchanged messages, ensuring that information can be interpreted correctly.

Semantics creates shared understanding: all agents give the same meaning to terms and concepts, using a common vocabulary that enables cooperation even between different technologies.

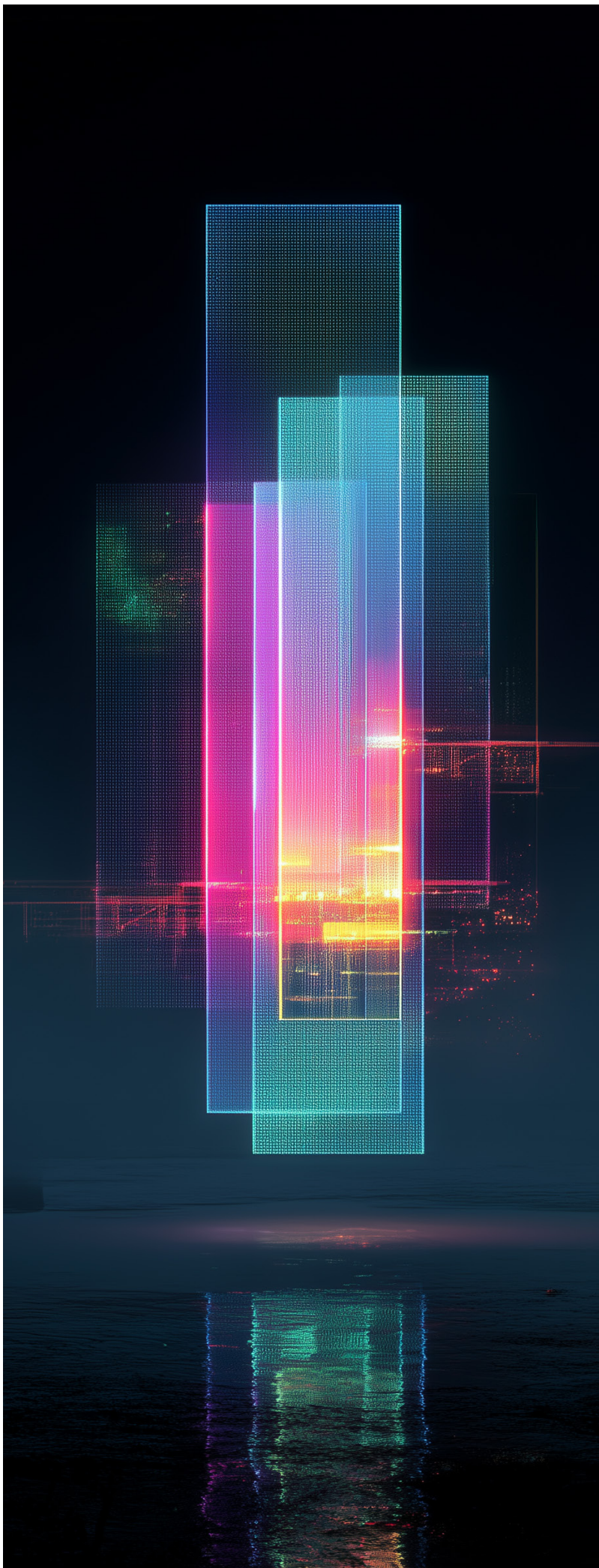
Pragmatics establishes the context, the rules of dialogue, and the intentions that guide interactions, allowing for an understanding not only of what is said, but also why it is said.

The rigorous adoption of integration and communication protocols is the necessary condition for enabling interoperability at the enterprise level and then on a vast scale, allowing agents developed by different teams, based on heterogeneous technologies, or belonging to distinct companies, to understand each other and collaborate effectively to create value greater than the sum of the individual parts.

The importance of integration and communication protocols and open standards is considerable: just as the Internet grew thanks to shared and interoperable protocols like TCP/IP and HTTP, multi-agent systems are also thriving where they are based on open and collaborative foundations. In this sense, open standards guarantee flexibility, avoid technological lock-in, and foster continuous innovation.

For companies, this means being able to choose modular, easily integrable, and scalable solutions, reducing the risks associated with adopting proprietary technologies in a high-speed market. Emerging standards that promote interoperability between different agents, such as the Model Context Protocol (MCP) and Agent2Agent (A2A), contribute to making this openness possible. Although they have distinct purposes, both aim to facilitate the construction of more robust and interoperable AI ecosystems.

MCP serves to connect agents with data sources, APIs, and contextual knowledge, facilitating access to external information in a structured and secure way. A2A is focused on collaboration between agents, enabling task delegation, the discovery of new capabilities, and the exchange of operational information. Thanks to these protocols, agents belonging to different vendors can communicate, collaborate, and coordinate as part of a single distributed system.



The Model Context Protocol for Standardised Context Management

The Model Context Protocol is an open-source project promoted by Anthropic, created to solve a frequent business problem in the adoption of LLMs: context management and the complexity of integration with heterogeneous tools. Before MCP, developers and companies found themselves having to build ad-hoc connectors for every possible combination of model and external service, generating a multiplication of costs and labour.

MCP instead proposes a universal interface that standardises typical operations such as reading and writing files, invoking remote functions, querying APIs, controlling complex software (e.g., 3D graphics applications like Blender), and even interacting with digital and physical systems. MCP allows agents to use the same interaction patterns regardless of the data source or the underlying LLM engine.

The protocol plays the role of a concrete link to operational reality: it defines a vocabulary, a grammar, and operational rules that transform the "cognitive state" of the model into something traceable and interpretable by the company's infrastructure. This anchoring process makes it possible to connect the abstract intelligence of the models to real resources (datasets, services, company APIs) and to maintain the consistency of the operational context across multiple agents.

In essence, MCP is certainly a technical integration mechanism, but also a conceptual framework that enables interoperability, transparency, and traceability in the behaviour of intelligent agents.

The protocol's architecture insists on a clear separation of roles between Host, Client, and Server. The Host acts as an orchestrator: it manages interactions with the user, applies company policies, merges contexts from multiple sources, and coordinates connections to the servers that expose functionalities. The Client instance embodies a single session or task: it negotiates the required capabilities, isolates the session state to prevent context leakage between different instances, and securely mediates between the Host's logic and what is offered by the Server. The latter, executable locally or remotely, exposes tools and resources in the form of standard primitives: APIs, database queries, domain functions, or encapsulated business logic, which are presented with metadata and capability contracts that the Client can negotiate and the Host can authorise.

The separation between Client and Server, in particular, is not just a technical choice; it combines business objectives with improved governance and a radical re-engineering of systems. Previously, the agent development lifecycle typically included both the logic of each agent and the development of the tools at its disposal, indissolubly linked in the same codebase. With MCP, the engineering of agents and that of services become two separate and decoupled problems.

This allows for the creation of teams specialised in transforming heterogeneous systems (software, databases, physical systems) into standardised and accessible services, and teams specialised in the engineering of the agents themselves, who can thus focus on memory, context management, and the application of filters and constraints on the available external services.

The Host/Client/Server "triad" defines practical operations such as capability negotiation, context merging, and the distinction between temporary session state and persistent context. The security model proposed by MCP is, consequently, distributed but coherently governed: it reflects principles of zero-trust and multi-tenancy and requires multi-level controls. At the Host level, policies define which tools and resources can be exposed and with what access policies; at the Client level, session isolation, information flow control, and prevention of cross-context leakage are applied; at the Server level, validation, authorisation, and compliance rules are imposed on the exposed data and operations.

However, the MCP ecosystem still faces several concrete risks: for example, prompt injection attacks that manipulate input to induce unexpected behaviours; permissions that are too broad, which could allow the exfiltration of sensitive files; the possibility that malicious "lookalike" tools could replace legitimate ones. Added to these is a crucial risk related to the governance of dependencies: the absence of rigorous server versioning.

Just as it is fundamental to define precise dependencies during software development, it is equally critical to establish which specific version of an MCP server an AI system must connect to. Without this rule, a malicious actor could modify a server, release a new compromised version that does not expose the expected tools or that hides malicious functionalities, and the AI system would use it without realising.

These potential vulnerabilities in MCP show how the autonomy and power of agents require countermeasures that go beyond simple channel encryption: what is needed are policies of least privilege, allowlisting mechanisms, explicit server versioning, signing and attestation of tool capabilities, validation of operator manifests, and continuous auditing of calls and decisions.

Risk mitigation: technical and organizational practices promoted by MCP

Explicit negotiation of tool capabilities between Client and Server

Validation of APIs and payloads before they are made interpretable by the model

Immutable recording of interactions for audit and forensics purposes

Adoption of runtime controls such as sandboxing, rate limiting, and behavioral anomaly detection

These measures must be integrated with corporate governance processes that include policy versioning, tool certification procedures, penetration tests specific to prompt-based attacks, and incident response plans that account for the autonomous nature of the agents.

Beyond the need to govern these potential complexities, the strategic advantages deriving from the adoption of MCP are manifold and linked to its ability to standardise the operational context. By allowing agents to adapt their behaviour in real-time based on updated information, without resorting to long and costly retraining cycles, the protocol improves the quality of decisions and the effectiveness of the actions taken. Architectural scalability is another practical result: exposing a new tool via an MCP Server makes it immediately available to all authorised instances, drastically reducing integration times and costs.

Added to this are the benefits of transparency and traceability, which facilitate compliance and auditability, making it simpler to demonstrate to stakeholders and regulators how and why an agent used certain information or decided to execute a specific action. Moreover, adoption in the field confirms the protocol's growing relevance: it is estimated that many thousands of MCP servers are active today, a sign of concrete and widespread interest.

For top management, it becomes clear that MCP, rather than being a mere technical connector, must be considered a critical infrastructure that requires investment in security, governance, and training. On the one hand, integrating MCP means reducing integration costs and accelerating time-to-market; however, it also imposes the definition of organisational responsibilities, continuous validation processes, and metrics to measure impact and risk.

Reply's experience shows that only with adequate rules and controls can the openness and democratisation offered by the protocol be transformed into real value, preventing the greater autonomy of systems from becoming a new attack surface. MCP can thus represent a fundamental step in evolving linguistic models from isolated text generators to cognitive partners fully integrated with the company's information ecosystem.

Its strength will grow over time, bringing order and rules to an otherwise chaotic landscape of custom integrations and providing the tools for scalable, secure, and traceable integration.

The Agent2Agent Protocol for Structured and Interoperable Collaboration

The Agent2Agent protocol, promoted by Google, represents an interesting initiative in defining an open standard for communication between intelligent agents. It is situated within the so-called "horizontal dimension" of artificial intelligence, which does not concern the training or internal capabilities of the single agent, but rather the possibility of enabling autonomous entities—developed by different providers, operating on heterogeneous platforms, and with different roles—to interact in an effective and structured way. In this context, A2A proposes itself as a sort of "lingua franca," a shared framework that allows agents to cooperate, exchange information, and coordinate in solving problems.

The central objective of A2A is to overcome the operational isolation in which AI agents often find themselves, fostering the emergence of ecosystems composed of "dynamic teams" in which each agent contributes its specific skills to the resolution of a collective task. The vision is that of a distributed market of abilities, where the interaction between agents is not rigid, but fluid, dynamically discovered based on operational needs. This approach makes it possible to tackle articulated tasks, which require different capabilities, in a scalable, resilient, and adaptive manner.

A2A Protocol: three guiding principles underlying the design

1. The first, "protocol-first," establishes that interoperability is guaranteed only through strict adherence to the shared specifications: participants must faithfully adhere to the protocol, avoiding customised solutions that would compromise compatibility.

2. The second principle, modularity, requires that every component of the system be built to be easily replaceable and composable, to support a future-proof architecture.

3. Finally, the concept of progressive improvement encourages gradual adoption: starting with simple solutions and introducing complexity only when necessary, according to the principles of agile development and iterative design.

The operational heart of A2A is the formalisation of communication between agents through structured messages. These messages can represent intentions, objectives, action plans, or results, and are transmitted through standard formats. The ability to exchange information in a semantically rich way allows agents to negotiate, delegate tasks, update the status of ongoing activities, and, potentially, learn from past experience and interactions. This structured messaging system, in addition to improving the transparency of the decision-making process, makes advanced forms of cooperation and coordination possible, even in open and distributed environments.

A key element of the A2A architecture is represented by the so-called Agent Cards. These are publicly accessible JSON files that describe in detail the agent's identity, its capabilities, the conditions of use, and the ways in which it can be engaged. These "digital identity cards" function as a discovery mechanism within the network of agents, allowing each participant to dynamically search for and select the most suitable collaborator for a given task. The most direct analogy is that of a company selecting a supplier based on its profile and credentials: A2A makes a similar dynamic possible, but fully automated.

The interaction between agents is structured around a flexible client-server model, in which roles can change dynamically over the course of the same conversation. An agent may initially act as a client, assigning a task, and subsequently become a server in a later phase or in a parallel context. Each activity passes through a defined lifecycle, which includes various states of progress (or suspension) of the activity and also supports intermediate updates for long-running activities. The output of a completed task is returned in the form of an artefact, a structured result designed to be easily processed by other agents or software systems. This approach guarantees coherence, traceability, and interoperability even in the most complex workflows.

The adoption of the A2A protocol is facilitated by the use of consolidated technologies such as HTTP, JSON-RPC, and Server-Sent Events: this significantly lowers the entry threshold for developers and companies interested in integrating the protocol into their systems. Agent Cards, following the standard, can be published and indexed on an accessible network, similar to the publication of services in a microservices architecture, fostering scalability and gradual integration.

A2A, in addition to being a technical specification, can be seen as an operating paradigm. Intelligence is no longer confined to the autonomous processing of the single agent, but is distributed throughout the system via interaction and cooperation. This change in perspective is notable: it moves from a model centred on individual capabilities to one that valorises the synergies and emerging resilience of the system as a whole. If one agent fails, for example, another with similar skills can take over, ensuring operational continuity and increasing overall reliability.

Within a few months of its launch, the protocol has been adopted by numerous large technology companies, such as SAP, Oracle, MongoDB, Atlassian, and PayPal. This broad support testifies to the interest in standardising collaboration between agents, especially in a context where companies find themselves integrating increasingly complex and distributed systems. Furthermore, being agnostic to the technology provider makes A2A particularly attractive for companies that want to avoid vendor lock-in and build flexible ecosystems, capable of evolving over time without being tied to a single platform.

A2A has the potential to be a good response to the growing need for coordination between autonomous entities in complex and constantly evolving environments. The value it produces lies less in enhancing the performance of the individual agent and more in the system's ability to respond in a cohesive, adaptive, and intelligent way to operational challenges of increasing complexity. Its adoption marks the transition from specialised, albeit powerful, models to a distributed collective intelligence, which draws strength precisely from the diversity and cooperation among its components.

A Complementary View of MCP and A2A

The combination of MCP and A2A can support a multi-agent architecture capable of overcoming many of the current limitations of distributed AI systems. These two protocols, while operating on distinct planes, reinforce each other and can be merged into an advanced architectural pattern. Together, they create an ecosystem in which the contextual capacity of individual agents and their ability to cooperate are significantly amplified. MCP provides agents with a deep and updated understanding of the operational context, while A2A allows that understanding to become collective, shared, and operationally useful at scale.

This complementarity translates into a dynamic where each agent accesses data and tools in real-time, but can also transmit inferences, states, and decisions to the other agents involved, allowing for fluid coordination on complex tasks. MCP acts as a lens through which the individual agent interprets the world, while A2A is the distributed neural network that allows multiple agents to compare notes, react, and plan together based on what has been observed.

By adopting MCP and A2A together, companies can move from adopting isolated AI systems to designing distributed organisations of cooperating agents, capable of adapting dynamically to evolving conditions. MCP allows agents to continuously update their understanding of the context in which they operate, but it is only thanks to A2A that this knowledge can be propagated, coordinated, and related to that of other agents.

The result is an emergent behaviour that goes beyond the sum of individual capabilities, where collective decisions are not only parallel but also interdependent and contextualised.

This synergy enables operational scenarios in which agents specialise but remain coordinated. For example, in an HR workflow, one agent can access candidate information and assess their compatibility, while a second manages the interview schedule, and a third handles the onboarding process. MCP guarantees each agent access to the relevant information, while A2A allows for the handoff of state and the coordination of actions. No agent works in isolation, and each contributes to a coordinated and continuous sequence of activities.

The same applies to domains like customer service, where responsiveness becomes essential: MCP allows agents to access CRM data and request details, while A2A facilitates the intelligent escalation of tickets, the recommendation of solutions by specialised agents, and the management of automated follow-ups. Each response to the customer is no longer the result of a single AI system, but the result of a collaboration between agents operating with updated and shared knowledge.

Another aspect where the complementarity between the two protocols strongly manifests is in the overall system's ability to evolve over time. MCP allows agents to update their strategy based on new information, while A2A allows these updates to spread among related agents. This makes the entire system capable of learning, adapting, and responding in a coordinated manner, even in the presence of unpredictable events or conditions of operational instability. It is a key transition from reactive logic to a dynamic, goal-oriented response capability.

From an architectural point of view, the interaction between MCP and A2A also allows for the decoupling of local decision-making logic and global coordination. This modular approach, combined with the open nature of the protocols, promotes true horizontal scalability, where new agents can be integrated without having to radically modify existing components. Instead of through centralised control, the system's coherence is maintained thanks to the sharing of context and the interoperability of agents, made possible precisely by the interaction between MCP and A2A.

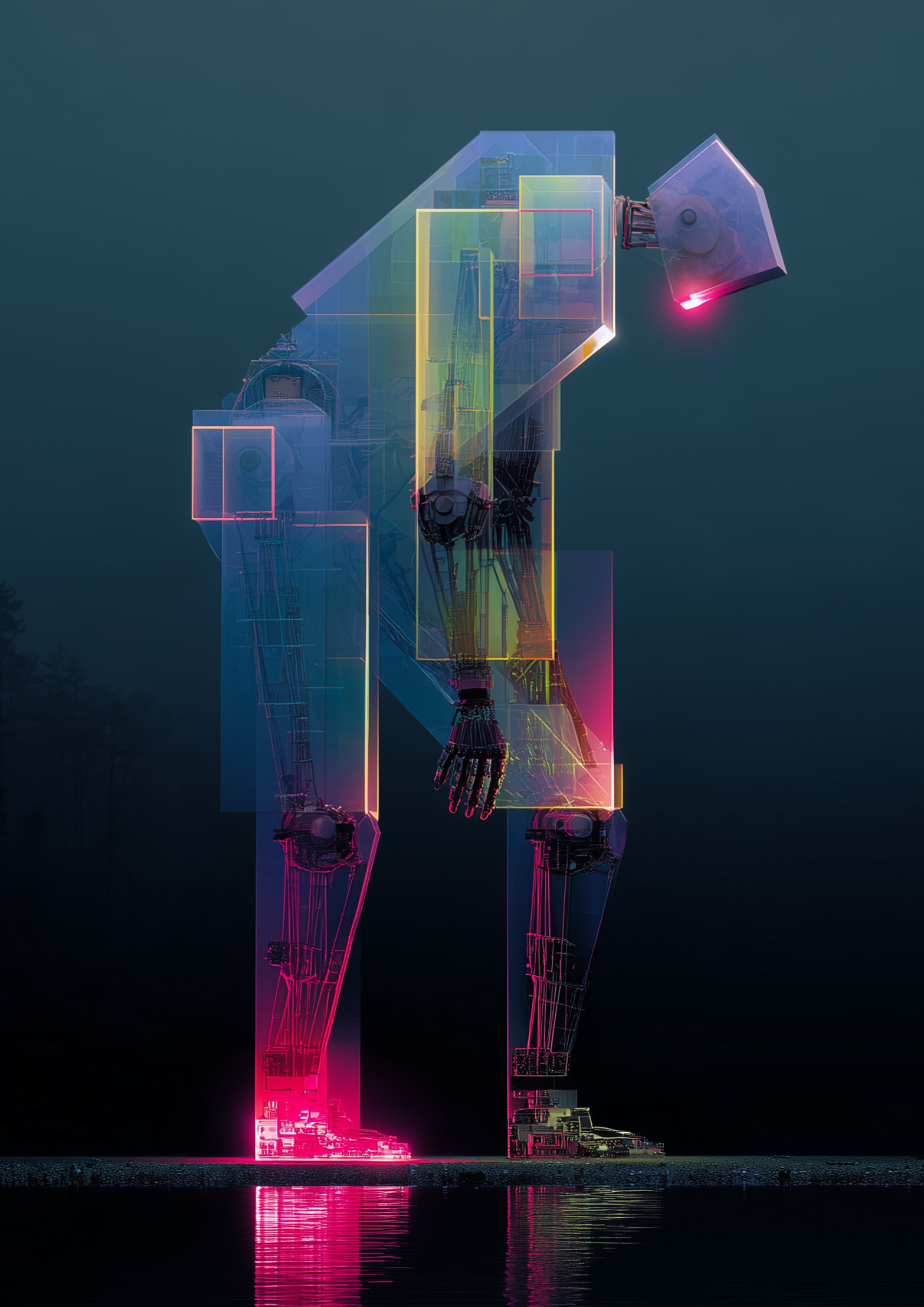
On a strategic level, this synergy paves the way for a new generation of business workflows: processes that are not only automated but also orchestrated, distributed, and capable of adapting to the constraints and opportunities of the context in real-time.

The benefits are cross-cutting: greater operational efficiency, reduction of manual interventions, faster response times, and decisions that reflect a deeper, shared understanding of the situation. The complementarity of the two protocols also proves to be an advantage on the governance and security front: MCP centralises access to context and guarantees semantic coherence, while A2A provides secure and standardised channels for information exchange between agents.

Together, they support the traceability of decisions, compliance with company policies, and adherence to regulatory requirements, helping to create a powerful, reliable, and potentially more compliant AI infrastructure.

In the medium term, the two protocols may be increasingly "merged" to create even more powerful and modular architectures. An entire system of agents, cooperating internally via the A2A protocol, could be exposed externally as a single MCP server. In this scenario, an external agent (or another organisation of agents) will not need to know the complexity of the internal collaboration; it will see the entire system as a single "tool" or specialised service to interact with.

This pattern will not only simplify integration but will also allow for the construction of hierarchies of AI systems, where orchestrator agents delegate complex tasks to specialised cooperating systems, making the overall architecture more scalable and maintainable.



3 The Reply AaaT Agent Network Experiment

A Framework for Orchestrating Distributed Intelligent Systems

The evolution of AI-based applications towards distributed ecosystems of specialised agents is often driven by the need to solve complex business problems that require different capabilities, access to heterogeneous data sources, and the execution of multi-stage decision-making processes. Even with the rapid evolution of protocols, at the enterprise level, the interaction and orchestration of a network of intelligent agents still introduce significant architectural challenges.

Managing communication, "discovering" capabilities, and assigning tasks can quickly become complex, limiting the scalability and reliability of the overall system.
It is in this context that Reply is experimenting with the AaaT (Agent as a Tool) Agent Network framework, conceived as a modular and scalable middleware layer, aimed at orchestrating interactions within a distributed network of intelligent agents.

AaaT was conceived as a way to define large-scale architectures with the same protocol-based philosophy as A2A, but without being bound by the latter's maturity or complexity.

A2A, in fact, is not just a protocol to be respected according to a precise signature, but a philosophy on how to make distributed AI systems interact. Since a subset of its specifications is often sufficient for a real system, AaaT implements its fundamental principles in a leaner, more pragmatic alternative, designed to be implemented effectively in concrete scenarios.

The AaaT Agent Network is based on a distributed multi-agent architecture, in which each agent is implemented as an independent service, thus promoting modularity, isolation, and scalability. The system is structured around three fundamental components.

AaaTServerNetwork: coordination and routing of agents

The first component is the AaaTServerNetwork, which acts as the central coordinator of the middleware. Its main responsibilities include dynamically registering agents as they become operational, managing their availability to other clients or agents, and acting as a proxy for direct calls to the appropriate agents. It maintains an in-memory registry that maps each agent's unique identifier to its deployment address.

Agents: autonomous and isolated services for flexibility and resilience

The second component is represented by the agents themselves. Each agent is an autonomous service, identified by its own unique code, technically implemented as an independent application, for example based on FastAPI; the determining factor is compliance with the protocol signature. The independence of each agent ensures that any failure is contained and does not spread to the entire network, and allows for great deployment flexibility.

AaaTClientNetwork: logic and communication via MCP

Finally, the AaaTClientNetwork is responsible for interpreting business logic based on user input and managing protocol-level communication with the central server to execute tasks. This communication is enhanced by the Model Context Protocol, which ensures structured and reliable information exchange across the network. In this way, the AaaT structure concretely implements the "fusion" pattern discussed above: the entire network of agents, orchestrated by the server, is exposed as a single MCP server, making it "usable" by other AI systems as if it were a single system.

The architectural design adopted for the AaaT Agent Network offers concrete advantages for the development and management of complex multi-agent systems. The isolation of each agent as an independent service guarantees fault containment and offers considerable deployment flexibility. The ability of agents to operate independently facilitates parallelism, allowing multiple tasks to be processed simultaneously, which translates into an increase in the overall productivity of the system.

This approach also encourages domain specialisation, making it possible to design agents optimised for very specific capabilities, thus achieving superior performance in each functional area. A further advantage is the loose coupling between components, made possible by the use of a central registry. This mechanism allows agents to be dynamically added, removed, or updated with minimal service interruption, ensuring the system's agility and adaptability over time.

Focus on Communication and Memory

Reply's experience has highlighted that the choice of communication paradigm is a critical factor for the scalability and robustness of multi-agent systems. Initially, the Reply AaaT Agent Network had adopted a synchronous communication model, but significant limitations emerged: relying on synchronous and persistent connections introduces fragility. Network latency accumulates throughout the system, and instabilities such as dropped connections or timeouts can cause deadlocks or cascading failures.

A fundamental distinction was therefore introduced between agents and simple "tools": the latter are typically stateless and designed for quick, isolated tasks; agents, by contrast, are conceived as autonomous, stateful entities, capable of managing complex, long-running operations that unfold over time. The introduction of a coordination middleware has centralised task delegation, ensuring that invocations are routed deterministically and precisely, eliminating ambiguity and conflicts.

Today, to ensure predictable and reliable interaction, every agent in the AaaT network must support a series of standardised commands: the protocol defines a common language for orchestration. An agent must always provide a mechanism to check its operational status, allowing the system to confirm its availability before sending a task. It must also offer a function to request a detailed description of its capabilities, functions, domains of expertise, and even interaction examples.

For execution, a specific command is used for the asynchronous submission of complex or long-running tasks. The lifecycle management of an active task is handled by commands for monitoring its status and for modifying or interrupting the activity.

The AaaTClientNetwork, which manages user interaction, implements a selective memory management mechanism to maintain the coherence and relevance of the conversation. Experimentation showed that forwarding the entire interaction history to the generative model led to a loss of focus and an increase in hallucinations after just a few complex interactions. To avoid this, the system passes only the most essential context to the model: the user's current question and the system's most recent responses.

This intentionally limited "short-term" memory is balanced by a network-level "long-term" memory, provided by a specific function.

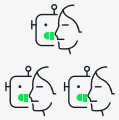
This capability is crucial because it allows the system to retrieve the status of all tasks previously started or still running on all agents. In this way, if a user resumes a session and asks for the status of a previous activity, the system can retrieve that information from the network, overcoming the contextual limitation of the conversational agent.

Initial Results of the Experiment

The evaluation of the AaaT system was conducted through the collection and qualitative analysis of communication traces generated during various experimental scenarios. One specific test evaluated the network's scalability and adherence to the protocol by increasing the number of operational agents.

SINGLE AGENTS

With a single agent, the system showed low latency.



THREE AGENTS

With three agents, the benefits of the coordinator's routing logic became apparent.

FIVE OR MORE AGENTS

With five or more agents, the advantages of asynchronous processing and task parallelism became significantly apparent.

During the tests, the basic task lifecycle management (submission and monitoring) remained robust at all experienced levels of complexity. The performance in advanced lifecycle management (retrieval of past tasks) showed a gradual, non-abrupt decrease as the number of agents increased, without ever reaching a breaking point within the tested range. The experience gained indicates that further modularisation could lead to a more scalable and maintainable system.

The AaaT Agent Network framework can be considered a solid model for building architectures based on intelligent agents. The use of a coordination middleware proved to be an effective approach for managing task delegation deterministically and avoiding resource conflicts; asynchronous, task-based communication was found to be fundamental for ensuring the system's robustness and scalability, unlike synchronous protocols, which introduce rigidity. The experimental results also confirmed that the architecture succeeds in maintaining protocol integrity and task management quality even as the network size increases.

The success of multi-agent architectures depends on interoperability and integration: shared standards, designed communication, and resilience by design are prerequisites for scalability and reliability.

4 Architectural Approaches and Lessons Learned

Rethinking the Foundations of Multi-Agent Systems

Interoperability remains the most complex challenge for multi-agent architectures. Without shared standards, the entire ecosystem risks collapse. Ontologies, meta-languages, and common formalisms thus become essential for building resilient environments, in which collaboration between agents can generate emergent behaviours superior to the sum of the individual performances.

In parallel, it is necessary to conceive a robust integration infrastructure. Well-documented APIs, standard interfaces, and reliable connectors constitute the starting point, enabling agents to access heterogeneous sources and interoperate stably with enterprise systems such as CRM, ERP, databases, or legacy platforms. For this reason, the design of gateways, middleware, and authentication and authorisation patterns must be initiated from the earliest phases of the technology roadmap.

Different architectural patterns are built upon these foundations. Centralised networks, based on a common knowledge base, offer uniformity and governance, but introduce risks of bottlenecks and single points of failure. Decentralised models offer robustness and scalability at the cost of greater complexity in ensuring convergence and global coherence. Hierarchical structures, with a leader agent coordinating specialists, are well-suited to complex processes like the supply chain, while horizontal ones foster collaboration and collective decisions, for example, in building investment portfolios. Then there are dynamic configurations, such as temporary coalitions and teams that form and dissolve based on objectives, offering considerable flexibility.

Communication between agents thus becomes a determining architectural choice. Explicit communication, via message exchange or requests, ensures precision and traceability, but can introduce latency. Implicit communication, based on observing the effects produced in the environment, is lighter but requires sophisticated inference mechanisms. In many scenarios, a hybrid approach prevails: critical components governed centrally, high-frequency processes distributed, all made possible by open standards that avoid lock-in and guarantee portability.

An operational transition plan must include resilience and continuity measures: distributed monitoring, circuit breakers to isolate faults, and alerting systems that make legal, financial, and reputational risks visible. Equally central is the management of memory and shared state. Agents must be able to retain information between sessions to improve contextualisation and accuracy, with exchange channels that may contain only final outcomes or, when required by regulations, also intermediate reasoning traces.

From the perspective of learning and adaptation, multi-agent architectures thrive in dynamic environments thanks to distributed learning and reinforcement mechanisms, which allow agents to update local policies in response to real-time feedback. However, Reply's experience shows that these mechanisms require controlled data pipelines, clear reward metrics, and strategies to avoid undesirable emergent behaviours, as well as offline validation procedures before release into production.

Specialisation and Governance

The quality and nature of the AI models that power the individual agents determine the overall capabilities of the system. Generalist, fine-tuned, or highly specialised models, together with techniques like Retrieval Augmented Generation (RAG), influence reliability, efficiency, and governance. Within the same company, some agents may orchestrate RAG requests to access external knowledge, others may rely on robust inferences obtained from fine-tuned models, and still others may operate on internal models subject to stringent controls.

In this scenario, roles become specialised: planner agents break down complex tasks into sub-activities, retriever agents select pertinent data, executor agents interact with APIs and operating systems, and supervisor agents monitor and validate output. This division allows for the construction of modular pipelines where each component is independent, updatable, and scalable, improving traceability and decision quality.

Modularity thus becomes a competitive advantage: isolating functionalities in specialised modules reduces complexity and limits the semantic inconsistencies typical of monolithic systems. For this to work, however, as we have seen, interoperability is essential: protocols and standardised data formats that allow different modules to be combined and heterogeneous components to be integrated without redesigning the stack.

To master this complexity, governance assumes a central role. The definition of roles and responsibilities, controls for data access and use, continuous tracking of decisions, and rollback planning are prerequisites for ensuring regulatory compliance and trust from stakeholders. These practices must be integrated with observability tools, distributed tracing, and disaster recovery plans to make scalability possible without compromising control.

Towards an “Internet of Agents”

AI-based architectures can achieve remarkable results, but they sometimes remain trapped in architectural and operational limitations that hinder their scalability and real-world use. Too often, agents live in closed ecosystems, designed to function within a single platform; this generates veritable islands of intelligence, incapable of interacting fluidly with third-party components. At the same time, communication and coordination mechanisms are often rigid and predefined: hard-wired flows, fixed state transitions, and closed protocols prevent dynamic adaptations and unexpected collaborations. Precisely those that are useful in real, distributed scenarios.

From these critical issues arises the concept of an Internet of Agents (IoA): a vast and interconnected network of heterogeneous agents that aims to transform isolated multi-agent systems into a distributed, resilient, and self-organising infrastructure. The analogy with the Internet is based on the fact that the IoA seeks to bring to agents the same benefits that the Web offered to people and companies, allowing for the collaborative composition of skills, the dynamic formation of teams, and the continuous discovery of resources. The objective is to move from closed sets of agents to a network in which each entity can publish its capabilities, discover others, join temporary groups for specific tasks, and, when needed, delegate portions of work to specialised sub-teams.

The operational heart of the IoA is the semantic registration and discovery of skills. For the network to become truly interoperable, agents must present rich and standardised descriptions of their capabilities: not simple tags, but structured representations that consider the domain of knowledge, accessible tools and APIs, confidence levels, performance metrics, and prerequisites. In addition to shared ontologies and vocabularies, mechanisms based on semantic embeddings and knowledge graphs can support the matching of complex requests with agent profiles, while ranking algorithms evaluate availability, expected latency, costs, and reliability history. Discovery thus becomes a multidimensional search, open to semantic matches and dynamic evaluations.

Group formation in the IoA is not necessarily monolithic: the network must support nested teams and delegation mechanisms. An agent that initiates a group may, during execution, detect the need for sub-problems to be entrusted to autonomous sub-teams, creating temporary hierarchical structures or horizontal collaboration networks. Communications within these groupings occur via structured messages that carry, in addition to the payload, essential elements and metadata such as conversation identifiers, timestamps, message type, sender, intended next recipient, and routing information. This approach supports orchestration, traceability, and auditing of interactions, also simplifying state recovery in the event of node failures or network interruptions.

The potential is tangible in concrete scenarios: in the collaborative writing of a scientific article, an agent assuming the role of coordinator can search for specialised writers, literature reviewers, and statistical validators, compose a group, divide tasks, and spawn sub-teams for specific analyses; in an industrial setting, a network of agents distributed across edge, cloud, and embedded devices can orchestrate sensors and actuators, optimise predictive maintenance processes, and adapt to real-world contexts without recompiling centralised software.

However, the challenges remain significant.

Team: metrics and feedback

Automatic team composition does not always guarantee the optimal combination of skills: effectiveness metrics, reputation systems, and feedback mechanisms are needed to enable the system to learn which configuration works best for which types of tasks.

Communication: less redundancy

Communication patterns can lead to information redundancy: message aggregation, semantic compression, and information coalescence policies become indispensable.

Scalability: optimized costs

The distributed nature could also potentially introduce significant computational and network costs: many agents that rely on large and expensive language models could make large-scale execution prohibitive if optimization mechanisms are not adopted.

To contain costs and make the IoA practical, it is useful to think about comprehensive solutions that include model-based and operational optimisations. The adoption of hybrid architectures, where lightweight models play the role of mediators, routers, or filters, and larger models are activated only when strictly necessary, can significantly reduce resource consumption. Techniques such as distillation, parameter-efficient fine-tuning, and semantic caching of frequent responses reduce the need for costly inferences. Batching protocols, message compression, and strategies for offloading to on-premises or edge can optimise latency and costs.

As with all AI projects, security and privacy represent fundamental design points. Strong authentication, execution sandboxing, resource access control, and end-to-end encryption for sensitive communications are prerequisites to prevent abuse and data leaks. The traceability of decisions via immutable logs and cryptographic signatures, together with mechanisms for verifying data origins, helps to establish accountability and audits. Rate-limiting systems, escalation policies, and conflict resolution procedures are necessary to manage anomalous behaviours or defective agents.

Techniques such as secure multiparty computation can enable cooperation that respects regulatory constraints or business needs.
From an engineering point of view, adopting a layered structure facilitates maintainability: improvements in the network infrastructure should not require redesigns in the interaction layer, just as the introduction of new storage mechanisms remains confined to the data layer.

This approach can allow for parallel development and the progressive integration of technological innovations, as well as simplifying testing, rolling upgrades, and incident management. Clear operational metrics (e.g., task latency, throughput, cost per task, success rate, and message repetition/redundancy rate) become indispensable for monitoring the health and utility of the system.

An Internet of Agents based on multi-agent architectures offers the prospect of a decentralised collective intelligence, capable of forming dynamically, interacting with the physical world, and responding to complex needs in a cooperative manner. The path towards real diffusion passes through both technical progress (ontologies, matching algorithms, semantic caching, model optimisations) and economic and regulatory choices that make the ecosystem sustainable and reliable.

If the Internet demonstrated the value of human interconnection, the Internet of Agents could represent the next step for multi-agent architectures: more than a simple technological extension, it is a platform to make the collaboration of autonomous entities possible, in a way that is as scalable as it is responsible.

5 Conclusions

Beyond the Architectural Impacts

The rise of multi-agent systems marks a significant step in the evolutionary path of artificial intelligence adoption by companies. The global success of agents based on generative AI is leading companies to focus on infrastructures capable of redefining organisational intelligence: instead of concentrating knowledge and decision-making power in a few nodes, AI offers the possibility of distributing them throughout the entire operational network, transforming the enterprise into a sort of widespread cognitive organism.

The value of this approach lies in the ability of multi-agent systems to generate emergent behaviours. While monolithic approaches tend to stiffen when faced with complexity, multi-agent systems transform it into a resource, orchestrating interactions between autonomous entities that learn, adapt, and collaborate. In this context, speed, rather than reactivity, becomes resilience, understood as the ability to reconfigure rapidly in the face of the unexpected and to transform the volatility of the context into an opportunity for growth.

For top management, this could mean rethinking coordination models. The traditional organisation, based on linear processes and vertical chains of command, should give way to a network of distributed intelligences that assist human teams, dialoguing in real-time. The highest-value decisions could indeed emerge from continuous interactions between humans and agents, according to a logic of co-decision. This is a profound change, which, in addition to improving productivity, requires a new managerial mindset.

Trust in less centralised models becomes the condition for unlocking the potential of MAS. It implies redesigning organisational roles and processes to allow professionals and agent systems to collaborate fluidly, building an internal narrative that accompanies people through this change. Moreover, the full adoption of a multi-agent paradigm invites a rethinking of the way of working, directly impacting processes.

Reply's experiences in various industries clearly show that MAS are no longer a laboratory prototype. From managing global supply chains to personalising customer interactions, from resource planning to dynamic risk monitoring, their value lies in the ability to integrate heterogeneous processes and information into a unique and cohesive ecosystem. This strategic strength of theirs goes beyond the local efficiency of a single function to hold complex systems together and make them evolve as a whole.

The Importance of Constant Control

In the medium term, multi-agent systems could become the underlying platform for new business architectures: Reply itself has already adopted multi-agent systems at the core of its corporate systems in key areas such as CRM and HR. As happened with cloud computing, MAS will increasingly become an essential enabling infrastructure: the companies that know how to anticipate this transition will not just be more efficient; they will build radically new business models that are more adaptive, faster, and more resilient.

However, adopting MAS so pervasively in enterprise contexts brings with it challenges proportional to their transformative power. Delegating operational (and even more so, decisional) capabilities to a network of autonomous agents necessitates the introduction of solid control and supervision mechanisms. Strong authentication, execution sandboxing, and the immutable traceability of interactions are non-negotiable conditions for guaranteeing accountability and security. The open nature of these architectures imposes a zero-trust approach, in which every interaction must be continuously verified and validated.

Another major challenge is that of economic and computational sustainability. The large-scale execution of agents based on complex language models entails significant costs. To ensure scalability, it is necessary to adopt sophisticated optimisation strategies: hybrid architectures that combine lightweight models for routine tasks with more powerful models activated only when essential; distillation techniques to compress knowledge and reduce consumption; and semantic caching to avoid redundant processing.

Furthermore, the intrinsic complexity of these systems requires a step change in the discipline of MLOps and observability. Reply's experience shows the need for advanced distributed tracing tools, multi-agent debugging practices, and incident management playbooks, capable of maintaining the stability of increasingly articulated ecosystems, supporting the intelligent orchestration of complex networks of interconnected agents.

The responsibility for this constant control will inevitably fall to the technological leadership of companies. It will be up to the senior leadership to choose whether to use MAS as tactical tools, destined to optimise existing processes, or to accept the challenge of constant and structured control to build upon them a cognitive infrastructure capable of supporting a lasting transformation. The decisions made today will help to develop each company's ability to innovate and lead the markets of tomorrow.

Disclaimer

Reply

Reply [EXM, STAR: REY, ISIN: IT0005282865] specialises in the design and implementation of solutions based on new communication channels and digital media. As a network of highly specialised companies, Reply supports major industrial groups in the telecom and media; industry and services; banking and insurance and public sectors in defining and developing business models enabled by the new paradigms of AI, cloud computing, digital media and the internet of things. Reply's services include: consulting, system integration and digital services. **www.reply.com**