2025

AI Agent Interoperability

APPLICATIONS IN INDUSTRY

SOYOYE, FIYIN

Contents

Introduction

Agent interoperability refers to the ability of autonomous and independent AI agents to communicate with one another, exchange information, and work together towards achieving shared goals. As AI advancement has sped up in recent years, anticipation has moved from just human-to-AI communication, and autonomous problem-solving to groups of agents collaborating (The AI Citizen, 2024).

The main impediments to this progress are legacy approaches to system interaction that do not address the unique communication needs of AI agents. This paper starts by examining those prior interface paradigms and their limitations. Next, it discusses the emerging communication paradigms that are surfacing to make agent interoperability a reality. And finally, in the landscape section, we take a look at adoption trends and opportunities across industries, as well as a simulation of agent interoperability using a fictional biotechnology company as a model.

Emerging Interface Paradigms

Before AI agents, modern software development centred around distributed and decentralized systems that communicated using rigid, rules-driven protocols. These protocols can be categorized into two broad groups according to how they handle interactivity between the communicating systems: request-response patterns and event-driven patterns. Understanding these two classifications helps to highlight the characteristics and limitations of the prevailing communication paradigms.

Request-Response Communication Patterns

Interfaces that follow this pattern allow for synchronous and directly connected communication in a point-to-point style through which requests and responses are exchanged between multiple systems in real-time (*An Architect's Guide to APIs: SOAP, REST, GraphQL, and gRPC*, n.d.). This pattern is most often used in scenarios where intercommunication is transactional, direct, and requiring of immediate response/confirmation as in the case of a user sending a request to a website asking for a specific action and receiving a single response. Implementers of this pattern usually require that each participating application know the required request/response command schema. The request invoker/client party tends to control the cadence of communication with the server/provider's role is to receive and respond to requests as they arrive.

Application User Interfaces (APIs) are the most common implementations of this communication pattern. Four API protocols that have seen widespread adoption over the years are SOAP, REST, gRPC, and GraphQL.

The Simple Object Access Protocol (also known as SOAP) was introduced in the late ‘90s as an XML-based protocol that allowed enterprise applications to send and receive standardized documents across different platforms and even programming languages(*SOAP in Web Services | Ramotion Agency*, n.d.). SOAP was known for its high degree of complexity and specific format requirements which made it a great fit for industries such as telecommunications, healthcare and financial technology. However, its use began to decline as Representational State Transfer (REST) APIs rose in popularity.

The REST architecture, first developed in the early 2000s, offered a simpler and more performant approach to stateless communication between software systems. It quickly became the de facto representative of API protocols. REST solved a lot of SOAP shortcomings by enabling the use of lighter weight messaging formats like JSON, HTML and even plain text, and optimizing for integration with modern browser-based web services (*Types of APIs | Types Of API Calls & REST API Protocol | Stoplight*, n.d.).
The 2010s has seen the development of several new API protocols, the most widely accepted of which have been gRPC and GraphQL. GraphQL was developed by Meta as an answer to REST’s problem of over fetching (e.g. getting all the fields for a user when you only need the “name” field) or under fetching (e.g. needing multiple requests to get a user’s information, as well as their associated posts). GraphQL allows the invoking client to specific exactly the data it wishes to receive, as well as the format it is in.

gRPC was developed by Google to make data transfer between software applications even faster and more efficient by encapsulating data in binary format (H Jeremy, n.d.). It is a bidirectional, asynchronous data exchange protocol that requires both sides of a target-client pair to have access to the same schema, while allowing multiple requests and responses to be sent across a single connection once opened. gRPC is said to be 7 times faster at receiving and 10 times faster at sending data than REST, although it is a much slower paradigm to implement due to lower levels of support across major frameworks.

Event-Driven Communication Patterns

Interfaces following this pattern follow a publish-subscribe model of interactivity. This model is asynchronous, with completely decoupled parties that operate under a producer-controlled cadence. Producers maintain little-to-no awareness of consumers and emit events/data that are ordered and stored in event logs/streams that consumers may react to on their own schedule. This communication pattern is best used when there is a large volume of continuously flowing data that multiple decoupled systems need access to.

Practitioners in this space tend to group event-driven communication patterns under two main categories: *message queues* and *event streams*. Message queues act as infrastructure that producers can send messages to for temporary storage until a consumer receives and processes it. There is no requirement for the producer or the eventual consumer to be active at the same time (*Message Queues vs Event Streams in System Design*, 11:17:04+00:00). The most used message queues are the open-source RabbitMQ, Amazon's Simple Queue Service (SQS) and Apache's ActiveMQ service.

While message queues are optimized for asynchronous, eventual message transfer, event streams are designed to handle a continuous flow of events meant to be processed in real time. Event streams are better suited for high volumes of data transfer than message queues and can also perform intermediary data processing and transformation before final delivery. Some of the most used event streaming frameworks are Kafka and Amazon's Kinesis (Shwe, 2023).

Limitations of the existing paradigms

Both request-response and event-driven communication patterns effectively handle message transfer for different types of use cases. However, they begin to break down when it comes to enabling autonomous AI agents to discover and interact with one another, as well as with software systems without human involvement. Some of the reasons for this are discussed below:

- **Human-centric complexity**
  APIs are essentially human-centric communication contracts that can hide a level of complexity that is difficult for an autonomous agent to comprehend. Humans can figure out what API documentation might indirectly imply or bring domain-specific knowledge of the multiple-step workflows that a complete data request entails in a way that AI agents just won't be able to do without very specific prompting. This effectively renders a lot of enterprise APIs unusable to them (Davies, 2025).
- **The paradox of choice** (*Why Can't I Just Use an API?*, n.d.)
  Before an AI agent can make a communication request, it needs to figure out which of the tens of endpoints in an average enterprise service's APIs is appropriate for its end goal. AI agents can find it quite difficult to make these sorts of choices in a confusing landscape in which there are no standardized protocols for how to approach and interact with different types of tooling

- **No context handling**
  Most existing communication interfaces are built on paradigms like REST which, at its core, is a stateless protocol that treats each request as an independent transaction. This is a major challenge for AI agents that need to preserve conversational continuity across multiple requests
- **High-volume traffic** (*AI Agent Is Hitting Your APIs - Are You Ready?*, 2025)
  AI agent traffic patterns can look very difficult from human-generated traffic and can bear semblance to malicious bot attacks even when legitimate. Existing communication and rate limiting frameworks can have a difficult time recognizing and appropriately handling AI agent requests
- **Security vulnerabilities** (*AI Agent Is Hitting Your APIs - Are You Ready?*, 2025)
  It is left to individual communication interfaces to figure out new security challenges such as making sure AI agents are really who they say they are and that they are truly authorized to act on behalf of specific users (*Why Can't I Just Use an API?*, n.d.). These interfaces also need to make sure AI agents can only access specific tools that the user they represent has permissions to.

New Paradigms

As the limitations of conventional communication patterns become more widely recognized, a standards competition has risen in which technology giants fight to be the originator of the one dominant protocol, leaving behind a highly fragmented and siloed landscape. This section discusses the most prominent of those protocols, with each approaching the problem of interoperability in a fundamentally different manner. The examined protocols will be grouped by the type of agentic interaction that they enable.

Agent-to-Tool

This category describes communication protocols whose main function is to enable the exchange of information and commands, retrieval of data, etc between agents and external tools and resources such as from outlook to X. The communication protocol that has gained the widest adoption by experts and most attention from enthusiasts (who combined represent a small percentage of potential users) under this category is the Model Context Protocol.

Model Context Protocol (MCP)

(Hou et al., 2025) describes MCP as an emerging open standard that defines a unified, bi-directional communication and dynamic discovery protocol between AI models and

external tools or resources. It was introduced in late 2024 by Anthropic as a means of addressing the problem of fragmented communication across distinct systems.
MCP addresses a few of the fundamental challenges of agent-to-tool interoperability.

The first pain point is the requirement for every tool or service to have a manual API connection for integration, and MCP solves this by offering a unified interface for agents to connect with all the tools in a software system. Another pain point centres around existing agent-to-tool integrations functioning as individual wrappers/plugins around APIs. These plugin solutions were tied to the organization (e.g. OpenAI or Zapier) that built them, extending the ecosystem fragmentation problem, while also only allowing for one-directional communication. MCP transcends this problem by being open-source, truly platform-agnostic, and supporting tool-to-agent events, notifications, and communications as well.

A third fundamental pain point facing agent-to-tool interoperability was that Retrieval-Augmented Generation systems (a key method through which AI models can remain current knowledge-wise without retraining) remained a passive unidirectional system of information retrieval. But with MCP, they can perform actions such as information updates under a unified framework.

MCP Architecture

MCP subscribes to the host-client-service architecture framework

- MCP Host: this is the application that runs the client and executes the AI-based tasks that are created. It is the environment that the user primarily interacts with e.g. Claude Desktop and Cursor IDE
- MCP Client: it sits within the host and maintains the communication linkage between the host and the server
- MCP Server(s): it does the work of exposing all the actual functionality that agents need. This functionality comes in 3 main capabilities: tools (allowing the invocation of external services and APIs), resources (exposing data to AI models), and prompts (exposing reusable action templates for e.g. workflow optimization)

Adoption and Challenges

MCP has seen some of the most adoption of any communication protocol targeting agent interoperability challenges. Leading AI companies such as OpenAI and Anthropic, as well as developer tool creators such as JetBrains and Replit have used MCP. However, overall adoption patterns by the larger technology professionals remain uneven and concentrated in prototypes and small community projects. Internet platforms acting as walled gardens adds a layer of friction to widespread adoption since they choose not to expose their MCP enabled services to the wider Internet, and potential security vulnerabilities compound this reticence. (Hou et al., 2025) groups the security attack types into four potential patterns: malicious developers, external attackers, malicious users, and general security flaws, each of which needs careful examination and handling before business implementation of the model context protocol.

Agent-to-Agent

Protocols under this grouping exist to enable communication among multiple agents, particularly regarding standardizing interoperability across different types of infrastructure, platforms, and systems. The two protocols that have come to dominate the discussion of the principles of direct communication between are the Agent-to-Agent protocol (A2A) and Agent Communication Protocol (ACP), while the Agent Network Protocol (ANP) acts as a framework at the network layer describing how agents can discover and route information between one another.

A2A

(Ehtesham et al., 2025) describes A2A as a protocol designed to allow peer-to-peer collaboration between distinct AI agents using Agent Cards which describe what the agent can do and how it should be called. A2A architecture consists of three core components: the user, client agent, and remote agent. The user acts as the originator of the request to the protocol, without necessarily understanding the underlying agent. The client agent receives the user's request, checks the Agent Cards of all available remote agents, and figures out the best one to assign the request. Finally, the remote agent carries out the given task, as directed by the client agent. Each remote agent advertises an Agent Card that details the skills (actionable capabilities) it offers.

The client and remote agents communicate with each other using Server-Sent Events (SSE) that establish persistent HTTP connections between both parties for real-time information streaming. Task communications all follow the JSON-RPC 2.0 specification.

ACP

In contrast to A2A's task-assignment objective, ACP has a broader focus on enabling general-purpose messaging and communication between agents. It has a similar architectural structure to A2A that consists of the client, server, and agent with the client acting as the originator of communication and the server as a converter of client requests into agent invocations, while the agent functions as the execution component.

ACP communication is HTTP-based and reliant on REST APIs. All submitted requests must be in the ACP-compliant format and standard. ACP enables a full range of communication styles from multi-part messages to session-based interactions, as well as both synchronous and streaming executions (Ehtesham et al., 2025)

The Merge

The ACP protocol was launched by IBM in March 2025, while A2A was introduced just a month later in April by Google. (Ai & Data, n.d.) In September 2025, it was announced that both protocols would merge and retain the A2A name because of obvious alignments in core goals, target focus (agent-to-agent communication), and technical challenges (Jankovics](https://dotsquarelab.com/team#vince_jankovics), 2025). This merge was also a result of the awareness of the problem of fragmentation in this new field, and a need to consolidate efforts, knowledge and standards.

Limitations of A2A

A2A has been praised extensively for its ability to enable peer-to-peer agent interaction while resolving previous issues of fragmentation and isolation in the industry. However, an industry survey by (Ray, 2025) in May 2025 showed some limitations in adoption, and a need for more community-driven expansion. (Louck et al., 2025) describes some of the trade-offs between security and efficiency that the A2A protocol makes, especially in the context of handling highly sensitive user information in payloads. It goes on to present some of these vulnerabilities including the absence of limitations in token lifetime, lack of strong customer authentication (SCA), lack of transparency and user consent, etc.

ANP

(Ehtesham et al., 2025) describes ANP as a common framework that enables secure, decentralized collaboration among AI agents across the open internet on a Truly peer-to-peer model. Unlike the likes of A2A which operates in a client-server architecture agents using ANP independently find, authenticate and collaborate with one another without a centralized registry or controller system in place.

ANP Architecture

ANP architecture has three distinct layers (*ANP Technical White Paper*, 2024)

- Identity and Secure Communication Layer
  The first layer of the protocol defines the standards that agents use to uniquely identify themselves to others. ANP makes use of HTTPS-hosted decentralized identifiers (DIDs) for this identification
- Meta-Protocol layer
  This next layer defines our requirements capabilities and collaborations are exchanged through natural language and our communication details are negotiated between agents.
- Application Protocol Layer

In this layer agents advertise their capabilities through the agents description module and make themselves available for discovery by exposing service entry points.

Adoption and Limitations

There isn't a lot of information about the adoption rates of ANP, but it seems to be quite limited, without evidence of a lot of real-world use cases. ANP has been praised for its AI native protocol negotiation standards which unfortunately do mean AI negotiation overhead for the systems that use it.

Application Landscape

Industries Ready for Agentic Interoperability

The nature of technology innovation is that every major transformational wave of advancement eventually disseminates and benefits every industry. A good example of this is how advancements now taken for granted such as search and email have fundamentally changed how humanity does business. However, this transformation does not usually take place at the same pace or offer the same level of benefits to every single industry.

AI agents can be expected to follow this same pattern of technology diffusion, and it is interesting to analyse what makes an industry more primed and ready to take advantage of not just automation & AI, but the concept of sophisticated multi-agent systems that must coordinate communication among one another. Some of the characteristics that especially differentiate these industries from those who are not as ready are described below:

- **Significant Coordination Requirements**: companies in these industries have operational workflows that require multiple steps across different teams or even organizations. Figuring out how to leverage interoperability to reduce process inefficiencies at handoff points both with and without human supervision could result in substantial productivity gains.
- **Fragmented Systems and Data**: these sectors usually function by making use of multiple software systems, usually legacy and each with its own data format requirements and means of interfacing. This makes any form of standardization with the older methods (APIs, etc) difficult to execute.
- **High Integration Overhead:** companies in these industries find that maintaining their suite of teams, software systems, and processes adds a significant overhead cost to operations. Adding in new products or external partners also means a subsequent round of difficult coordination and additional costs.

- **External Requirements:** beyond all the aforementioned characteristics and challenges, these industries usually have strict regulatory constraints, as well as external requirements that add layers of complexity to their processes in terms of how data can be stored, used, accessed, and so on (*Seizing the Agentic AI Advantage | McKinsey*, n.d.)

AI Adoption Trends

In the past couple of years, technology transformation discussions have centred on how companies can begin to extract business value from the recent advancements of GenAI and AI agents. This section explores the status of AI adoption across industries sourced from surveys by leading organizations.

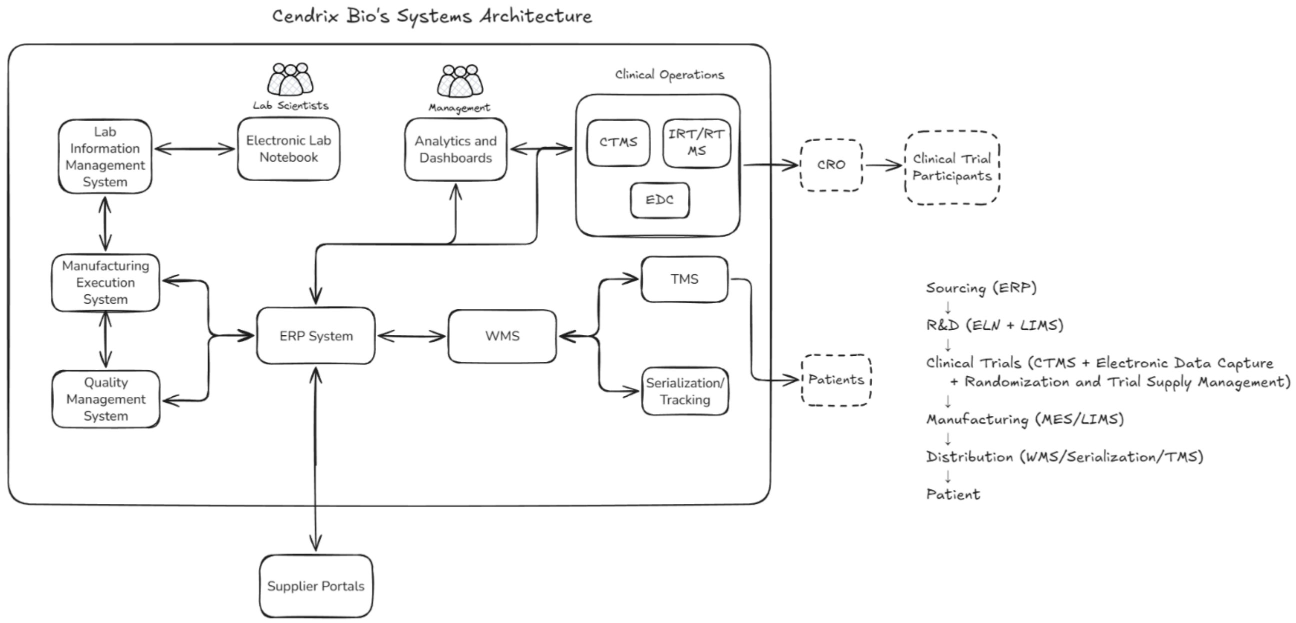
A survey from McKinsey in late 2025 showed that there was generally a lot of curiosity about AI agents with 67% of survey responses showing that their companies were experimenting with agents even though two-thirds admitted that they had not begun to scale AI across the organization(*The State of AI: Global Survey 2025 | McKinsey*, n.d.). PWC's 2025 survey showed quite similar trends with 88% of surveyed executives saying that their companies planned to increase their AI-related budgets and two-thirds of those using AI agents already reported increase in productivity (PricewaterhouseCoopers, n.d.). However, less than half of those adopting agents were rethinking and redesigning their operating models around AI. EY's equivalent survey showed that technology companies still led the charge on AI adoption and investment (organization|authorurl:https://www.ey.com/en_us/people/ey, n.d.).

McKinsey's in-depth analysis into advanced industries showed that a growing number of companies in complex and operations-intensive industries such as manufacturing and logistics have begun to use and see value from agentic AI. The top three drivers of this development have been automation, quality control & safety, and innovation. McKinsey's research also shows that agentic AI could result in a 5 to 10% increase in revenue (about 450 to 650 billion dollars) by 2030 for advanced industries with more gains also coming from cost savings (*Agentic AI Implementations in Advanced Industries | McKinsey*, n.d.).

Biotechnology Simulation

To make the discussion of the applications of AI more tangible, this subsection describes the software architecture of a fictional biotechnology company (called Cendrix Bio) and examines where agentic interoperability could have a positive impact on coordination difficulties and integration overhead in the company's operations.

Existing Architecture


Cendrix Bio's Systems Architecture
Lab Scientists
Management
Clinical Operations
Lab Information Management System
Electronic Lab Notebook
Analytics and Dashboards
CTMS
IRT/RTMS
EDC
CRO
Clinical Trial Participants
Manufacturing Execution System
Quality Management System
ERP System
WMS
TMS
Serialization/ Tracking
Patients
Supplier Portals
Sourcing (ERP)
↓
R&D (ELN + LIMS)
↓
Clinical Trials (CTMS + Electronic Data Capture
+ Randomization and Trial Supply Management)
↓
Manufacturing (MES/LIMS)
↓
Distribution (WMS/Serialization/TMS)
↓
Patient

Cendrix Bio is a relatively young and innovative biotechnology company that has had a couple of successful drug launches in recent years, mostly focused on gene therapy development. Their current operations are US-focused, although they make use of starting materials from other countries. They primarily outsource manufacturing to reputable Contract Manufacturing Companies (CMOs) but take on their fair share of internal manufacturing to support their clinical trials operations. They would like to expand their internal manufacturing capabilities in the future, but only after refining their existing structure and operations.

Outlined below is a description of their current systems architecture, grouped into four main subsystems:

Clinical Operations: the platform stack in this subsystem consists of the Clinical Trials Management System (CTMS), Interactive Response Technology/ Randomization and Trial Management System (IRT/RTMS), and Electronic Data Capture (EDC) systems. The CTMS is responsible for all the logistical and operational management of clinical trials, from study planning to trial site management and monitoring. The IRT/RTMS systems facilitate the randomization of trial participants and managing trial supply while the EDC is used to capture and manage participant trial data. This system usually interfaces with Contract Research Organizations (CROs) who are external organizations that handle participant enrolment, screening and overall trial management for biotech companies.

Supply and Logistics: This consists of the Warehouse Management System (WMS), Transport Management System (TMS), and Serialization & Tracking System. The WMS is an important component that takes care of the storage of product inventory while the TMS is responsible for its transportation, with Serialization & Tracking ensuring that each unit of inventory is uniquely identified and its movement tracked.

Manufacturing and Quality: The Manufacturing Execution System (MES) manages the production and manufacturing operations for Cendrix's drugs while the Quality Management System (QMS) tracks the manufacturing processes to ensure they are in compliance with quality and regulatory requirements. The manufacturing end of the Laboratory Information Management System (LIMS) manages the interaction of laboratory samples and workflows with the company's production process.

Research & Development: This section consists of the Electronic Laboratory Notebook (ELN), the research side of the Laboratory Information Management System (LIMS), and other miscellaneous project management and data analytics software. The ELN handles the documentation of experiments in the research process while the LIMS handles lab samples and testing procedures.

The Enterprise Resource Planning (ERP) is at the core of this system, sitting between all four subsections and acting as the authoritative source of truth and integrating supply chain, procurement, HR, R&D, financials and other sectors. In the current architecture, integrated workflows are driven by humans who need to create and export reports from individual systems, import them into others, create requests across platforms and receive information from email and, for example, Slack communication.

New Architecture

My approach to improving the operational bottlenecks described above is to incorporate a layer of context-specific AI agents that each manage their own domain and access to data and resources, while also communicating with one another to achieve common goals.

I will be focusing specifically on the section of Cendrix's architecture that covers Cendrix's clinical trials operations: from internal manufacturing to the storage and transport of medication to trial sites to interfacing with trial sites through the external Contract Research Organization (CRO).

The AI agents are described below:

The Contract Research Organization (CRO) Interface Agent: the role of the CRO interface agent is to sit within the CTMS, IRT/RTMS, EDC and other supporting systems to understand all trial operational data such as study and site startups, clinical trial enrolment data, patient screening, patient drop-out rates, etc. It regularly queries information from these systems through the MCP protocol while also communicating relevant information via A2A to other agents, particularly the Clinical Ops agent.

Clinical Ops Agent: the role of this agent is to manage the supply chain planning and forecasting for each individual clinical trial site. It tracks inventory data from the CTMS and ERP via MCP, receives enrolment data from the CRO interface agent and production status information from the MES agent before integrating all the data to decide whether to request new production batches or new shipments of trial medications. It communicates with the CRO interface, MES and Logistics agents via A2A.

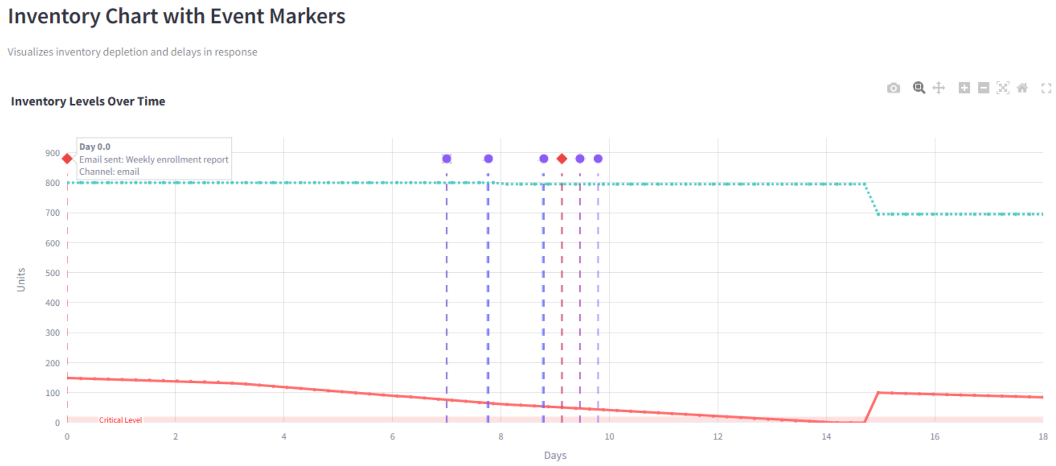
MES Agent: This agent sits within the MES, LIMS, and QMS systems and receives input from the Clinical Ops agent to decide when new batches of medication should be produced. It also tracks quality constraints such as temperature, expiration, faults and other regulatory requirements and reports issues back to the Clinical Ops agent. It informs the logistics agent of expected production and potential shipment dates.

Logistics Agent: This agent is responsible for coordinating with the TMS, LIMS and Tracking System, as well as the Clinical Ops and MES agents to determine how and when shipments should be made to clinical trial sites. It also handles the appropriate storage, identification/serialization and status tracking of each unit to be moved.

Scenario: Enrolment Spike at Trial Site [Mass General Brigham] (Us-BOS-001)

Link to simulation

To show the difference between Cendrix's system performance before and after introducing AI agents, I am describing a scenario in which one of Cendrix's clinical trial sites (out of hundreds of sites) experiences an unexpected spike in patient enrolment leading to a higher need for trial medication than forecasted. I created a simulation linked to illustrate the scenario:



Inventory Chart with Event Markers

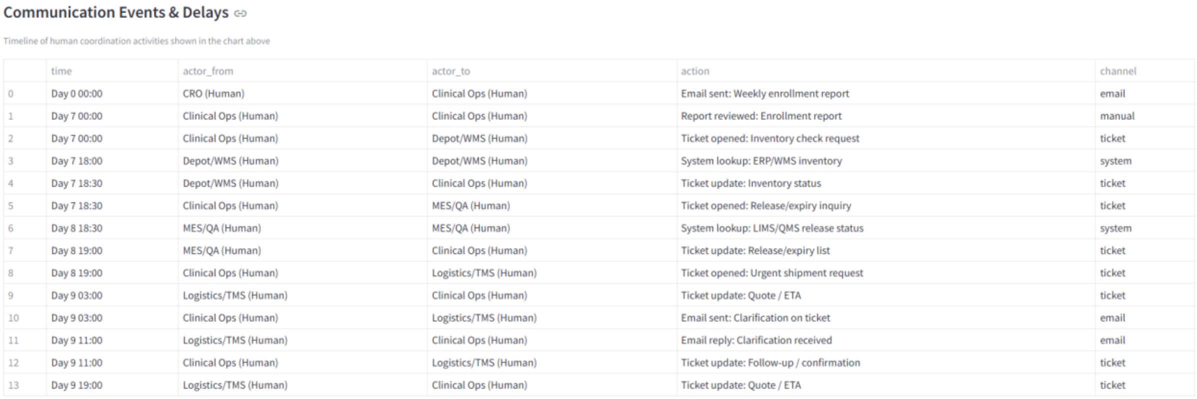
Visualizes inventory depletion and delays in response

Inventory Levels Over Time

Day 0.0
Email sent: Weekly enrollment report
Channel: email

Exhibit 1: Inventory levels over time for Cendrix's existing architecture

With the original system's architecture, it takes a week for a scheduled enrolment report to make it from the CRO to Cendrix's supply manager who notices the spike in participants and opens an inventory check request. It then takes an additional three days (and a total of nine days from the initial enrolment spike) to trigger an urgent shipment of medication to the affected site. The bulk of this time is taken up by waits and inefficient human handoffs leading to workflow inefficiencies that increase the risk of a stock-out as shown in exhibit 2 below:



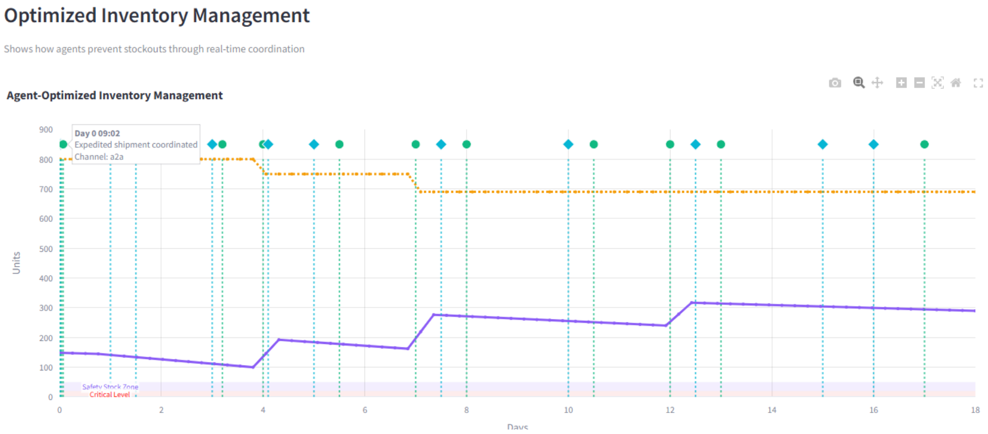
Communication Events & Delays

Timeline of human coordination activities shown in the chart above

	time	actor_from	actor_to	action	channel
0	Day 0 00:00	CRO (Human)	Clinical Ops (Human)	Email sent: Weekly enrollment report	email
1	Day 7 00:00	Clinical Ops (Human)	Clinical Ops (Human)	Report reviewed: Enrollment report	manual
2	Day 7 00:00	Clinical Ops (Human)	Depot/WMS (Human)	Ticket opened: Inventory check request	ticket
3	Day 7 18:00	Depot/WMS (Human)	Depot/WMS (Human)	System lookup: ERP/WMS inventory	system
4	Day 7 18:30	Depot/WMS (Human)	Clinical Ops (Human)	Ticket update: Inventory status	ticket
5	Day 7 18:30	Clinical Ops (Human)	MES/QA (Human)	Ticket opened: Release/expiry inquiry	ticket
6	Day 8 18:30	MES/QA (Human)	MES/QA (Human)	System lookup: LIMS/QMS release status	system
7	Day 8 19:00	MES/QA (Human)	Clinical Ops (Human)	Ticket update: Release/expiry list	ticket
8	Day 8 19:00	Clinical Ops (Human)	Logistics/TMS (Human)	Ticket opened: Urgent shipment request	ticket
9	Day 9 03:00	Logistics/TMS (Human)	Clinical Ops (Human)	Ticket update: Quote / ETA	ticket
10	Day 9 03:00	Clinical Ops (Human)	Logistics/TMS (Human)	Email sent: Clarification on ticket	email
11	Day 9 11:00	Logistics/TMS (Human)	Clinical Ops (Human)	Email reply: Clarification received	email
12	Day 9 11:00	Clinical Ops (Human)	Logistics/TMS (Human)	Ticket update: Follow-up / confirmation	ticket
13	Day 9 19:00	Logistics/TMS (Human)	Clinical Ops (Human)	Ticket update: Quote / ETA	ticket

Exhibit 2: Communication events for Cendrix's existing architecture

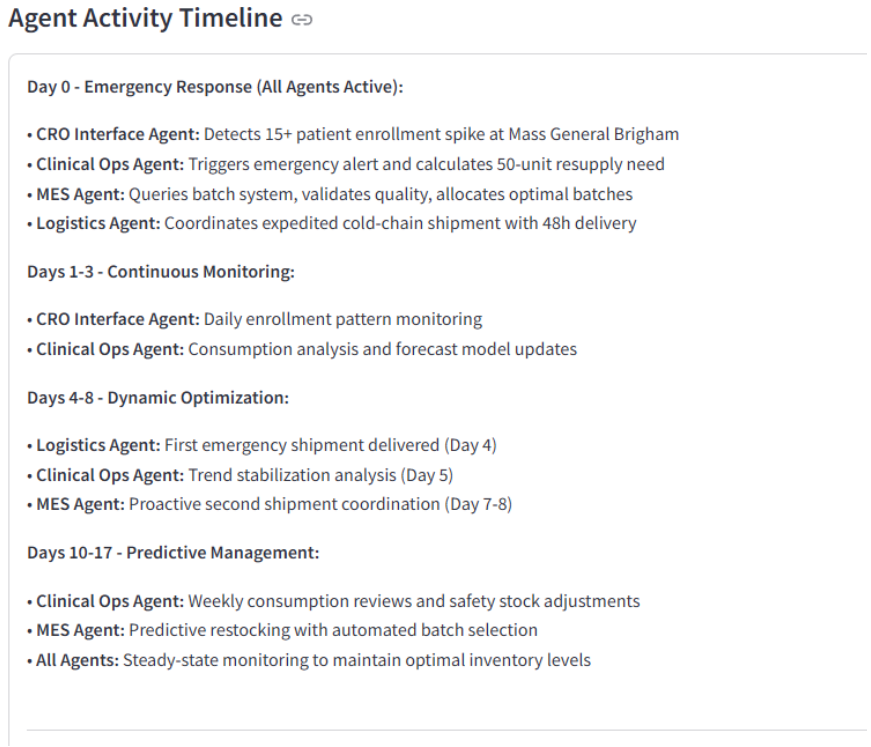
With the agent-optimized architecture, the CRO Interface agent performs real-time monitoring of patient enrolment levels across all sites. Once it notices the sudden increase in enrolment, it communicates with the Clinical Ops, MES, and Logistics agents to query current medication inventory levels at Cendrix, immediately allocate a batch for emergency resupply, fulfil all regulatory requirements, and trigger a shipment to Mass General Brigham, all within a few minutes and preventing a stock-out as shown in exhibits 3 and 4 below.



Optimized Inventory Management

Shows how agents prevent stockouts through real-time coordination

Exhibit 3: Agent-optimized inventory management



Agent Activity Timeline

Day 0 - Emergency Response (All Agents Active):

- **CRO Interface Agent:** Detects 15+ patient enrollment spike at Mass General Brigham
- **Clinical Ops Agent:** Triggers emergency alert and calculates 50-unit resupply need
- **MES Agent:** Queries batch system, validates quality, allocates optimal batches
- **Logistics Agent:** Coordinates expedited cold-chain shipment with 48h delivery

Days 1-3 - Continuous Monitoring:

- **CRO Interface Agent:** Daily enrollment pattern monitoring
- **Clinical Ops Agent:** Consumption analysis and forecast model updates

Days 4-8 - Dynamic Optimization:

- **Logistics Agent:** First emergency shipment delivered (Day 4)
- **Clinical Ops Agent:** Trend stabilization analysis (Day 5)
- **MES Agent:** Proactive second shipment coordination (Day 7-8)

Days 10-17 - Predictive Management:

- **Clinical Ops Agent:** Weekly consumption reviews and safety stock adjustments
- **MES Agent:** Predictive restocking with automated batch selection
- **All Agents:** Steady-state monitoring to maintain optimal inventory levels

Exhibit 4: Agent-optimized workflow timeline

Instead of waiting to receive weekly reports or emails and other manual communications, each agent is always available and subscribed to events affecting its domain. Whenever an anomaly is detected, they act autonomously to coordinate a response, while also bringing a human into the loop for costly and judgment-sensitive scenarios.

Conclusion

Recent changes in the technology landscape, especially advances in automation, AI and language models, have left entire industries grappling with how to practically make use of the new opportunities for streamlining complex workflows. This paper examines AI agents as a key capacity enabler not just in the context of independent, solitary work, but with collaboration and interoperability between agents as the focus. Using a simulated scenario in a fictional biotechnology company, it provides an illustrative example of the type of performance gains that companies can achieve given thoughtful and effective application of technology.

Further opportunities for research could include examining multi-company agent interoperability, as well as the integration of macroeconomic indicators such as trade policies and tariff schedules into supply chains. Another opportunity could be the exploring the end-to-end integration of AI agents across business areas from the ERP to finance, HR, and so on.

Bibliography

Agentic AI implementations in advanced industries | McKinsey. (n.d.). Retrieved 5 December 2025, from https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/empowering-advanced-industries-with-agentic-ai

AI Agent Is Hitting Your APIs—Are You Ready? (2025, August 1). Speedscale. https://speedscale.com/blog/ai-agent-is-hitting-your-apis-are-you-ready/

An architect's guide to APIs: SOAP, REST, GraphQL, and gRPC. (n.d.). Retrieved 30 October 2025, from https://www.redhat.com/en/blog/apis-soap-rest-graphql-grpc

ANP Technical White Paper. (2024, December 12). https://agentnetworkprotocol.com/en/specs/01-agentnetworkprotocol-technical-white-paper/

Davies, M. (2025, October 3). *AI Week: What Autonomous Agents Actually Need from Your APIs | Zuplo Blog*. Zuplo. https://zuplo.com/blog/what-autonomous-agents-actually-need-from-your-apis

Ehtesham, A., Singh, A., Gupta, G. K., & Kumar, S. (2025). *A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)* (No. arXiv:2505.02279). arXiv. https://doi.org/10.48550/arXiv.2505.02279

H Jeremy, J. (n.d.). *gRPC vs. REST: Key Similarities and Differences*. Retrieved 30 October 2025, from https://blog.dreamfactory.com/grpc-vs-rest-how-does-grpc-compare-with-traditional-rest-apis

Louck, Y., Stulman, A., & Dvir, A. (2025). *Improving Google A2A Protocol: Protecting Sensitive Data and Mitigating Unintended Harms in Multi-Agent Systems* (No. arXiv:2505.12490). arXiv. https://doi.org/10.48550/arXiv.2505.12490

Message Queues vs Event Streams in System Design. (11:17:04+00:00). GeeksforGeeks. https://www.geeksforgeeks.org/system-design/message-queues-vs-event-streams-in-system-design/

OpenAI. (2025). *ChatGPT (GPT-5.1 model; used for structuring, brainstorming and finding relevant papers to cite) [Large language model]. Https://chat.openai.com* [Computer software].

organization|authorurl:https://www.ey.com/en_us/people/ey, authorsalutation:|authorfirstname:EY|authorlastname:Americas|authorjobtitle: Multidisciplinary professional services. (n.d.). *EY survey reveals that technology companies are setting the pace of agentic AI – will others follow suit?* Retrieved 12 December 2025, from https://www.ey.com/en_us/newsroom/2025/05/ey-survey-reveals-that-technology-companies-are-setting-the-pace-of-agentic-ai-will-others-follow-suit

PricewaterhouseCoopers. (n.d.). *PwC's AI Agent Survey*. PwC. Retrieved 5 December 2025, from https://www.pwc.com/us/en/tech-effect/ai-analytics/ai-agent-survey.html

Ray, P. P. (2025). *A Review on Agent-to-Agent Protocol: Concept, State-of-the-art, Challenges and Future Directions*. Preprints. https://doi.org/10.36227/techrxiv.174612014.42157096/v1

Seizing the agentic AI advantage | McKinsey. (n.d.). Retrieved 12 December 2025, from https://www.mckinsey.com/capabilities/quantumblack/our-insights/seizing-the-agentic-ai-advantage

Shwe, T. (2023, December 22). Choosing Between Message Queues and Event Streams. *The New Stack*. https://thenewstack.io/choosing-between-message-queues-and-event-streams/

SOAP in Web Services | Ramotion Agency. (n.d.). Retrieved 30 October 2025, from https://www.ramotion.com/blog/soap-in-web-services/

The AI Citizen, T. A. C. by W. A. (2024, April 21). *What is AI Agents Interoperability?* The AI Citizen. https://theaicitizen.com/p/what-is-ai-agents-interoperability

The State of AI: Global Survey 2025 | McKinsey. (n.d.). Retrieved 12 December 2025, from https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai

Types of APIs | Types Of API Calls & REST API Protocol | Stoplight. (n.d.). Retrieved 30 October 2025, from https://stoplight.io/api-types

Why Can't I Just Use an API? Because Your AI Agent Needs MCP. (n.d.). Auth0 - Blog. Retrieved 30 October 2025, from https://auth0.com/blog/mcp-vs-api/