# 🦸 #14: What Is MCP, and Why Is Everyone – Suddenly!– Talking About It?

Community Article Published March 17, 2025

Upvote 32 +26

Kseniase
Ksenia Se

everything you need to know about Model Context Protocol

"Even the most sophisticated models are constrained by their isolation from data – trapped behind information silos and legacy systems." Anthropic, on why context integration matters

Large language models (LLMs) today are incredibly smart in a vacuum, but they struggle once they need information beyond what's in their frozen training data. For AI agents to be truly useful, they must access the right context at the right time – whether that's your files, knowledge bases, or tools – and even take actions like updating a document or sending an email based on that context. Historically, connecting an AI model to all these external sources has been a messy, ad-hoc affair. Developers had to write custom code or use specialized plugins for each data source or API. This made "wire together" integrations brittle and hard to scale.

To simplify that, Anthropic came up with Model Context Protocol (MCP) – an open standard designed to bridge AI assistants with the world of data and tools, to plug in many different sources of context. They announced it in November 2024. The reaction was sort of blah. But now MCP is trending, already passing Langchain and promising to overcome OpenAPI and CrewAI pretty soon. Major AI players and open-source communities are rallying around MCP, seeing it as a potential game-changer for building agentic AI systems. Why?

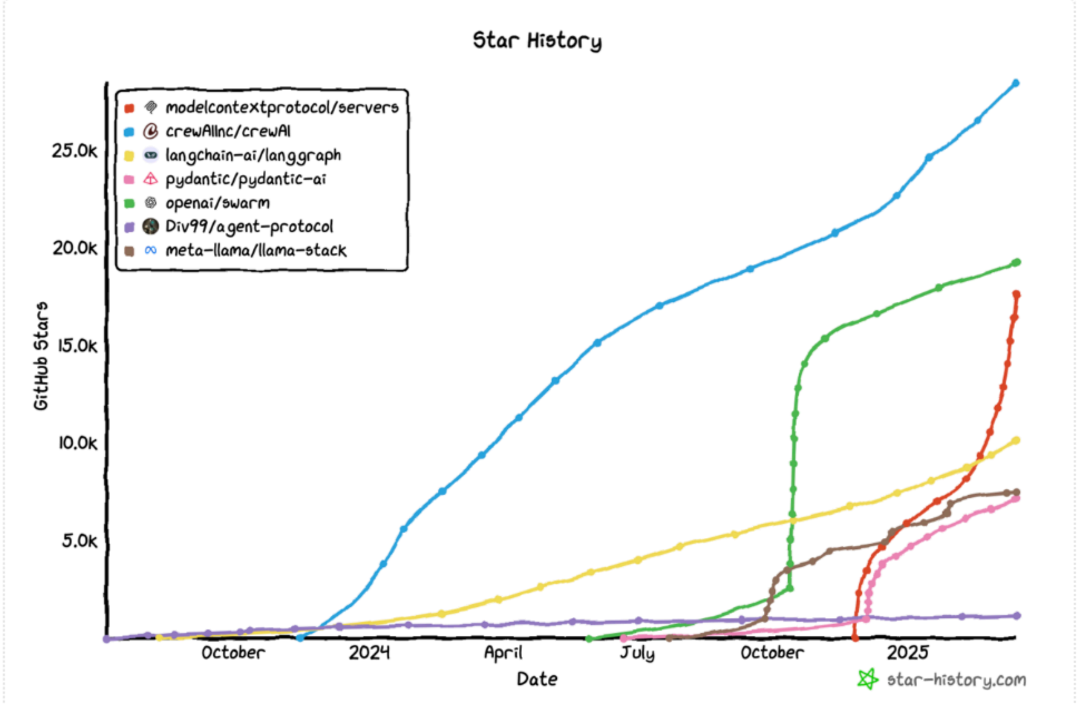


Image Credit: star-history.com

In this article, we'll dive deep into MCP – why it's a hot topic right now, how MCP enables the shift toward more integrated, context-aware AI, its place in agentic workflows, and the under-the-radar details that developers, researchers, AI engineers, and tech executives should know. We'll also explore some innovative applications of MCP that few have attempted. Overall, it's a great starting guide, but also useful for those who have already experimented with MCP and want to learn more. Dive in!

◼ Turing Post is on 🤗 Hugging Face as a resident -> click to follow!

What's in today's episode?

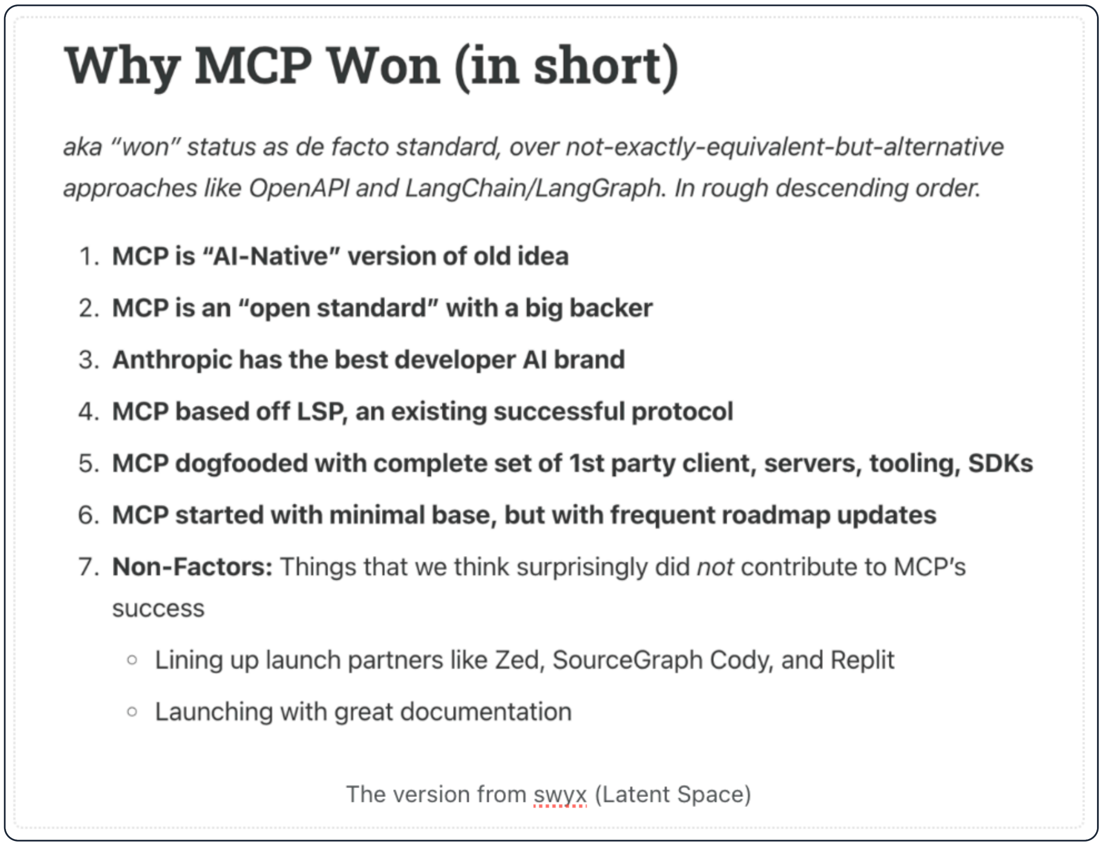
Why Is MCP Making Waves Now (and Not Last November)?

MCP was first open-sourced and announced by Anthropic in late November 2024. At the time, it was an exciting idea but not that many noticed it and took seriously. It's in early 2025 that MCP has really surged into the AI community's consciousness. There are a few big reasons for this recent buzz:

- **Integration Problem Solver:** AI agents and agentic workflows became major buzzwords in 2023–2024, but their Achilles' heel remained: integrating these agents with real-world business systems and data. Initially, much attention went to model capabilities and prompt techniques, not integration. MCP squarely addresses this gap by defining "how to connect existing data sources" (file systems, databases, APIs, etc.) into AI workflows. As people digested this, MCP started to be seen as the missing puzzle piece for serious, production-ready AI agents. (That's one of the takes from HumanX conference: In recent years, we've primarily been focused on building individual AI models, each specialized for specific tasks. But as complexity and demands grow, a shift is happening towards integrated systems – orchestrations of multiple specialized models, software components, APIs, data sources, and interfaces working cohesively.)
- **Community and Adoption:** In just a few months, MCP went from concept to a growing ecosystem. Early adopters included companies like Block (Square), Apollo, Zed, Replit, Codeium, and Sourcegraph, who began integrating MCP to enhance their platforms. Fast forward to 2025, and the ecosystem has exploded

– by February, there were over 1,000 community-built MCP servers (connectors) available. Clearly, MCP has struck a chord as the industry moves toward more integrated and context-aware AI. This network effect makes MCP even more attractive: the more tools available via MCP, the more useful it is to adopt the standard.

- **De Facto Standard Momentum:** Unlike yet another proprietary SDK or one-off framework, MCP is open and model-agnostic, and it's backed by a major AI player. This means any AI model (Claude, GPT-4, open-source LLMs, etc.) can use MCP, and any developer or company can create an MCP integration without permission. Many in the community now see MCP as the likely winner in the race to standardize how AI systems connect to external data (much like how USB, HTTP, or ODBC became ubiquitous standards in their domains).
- **Rapid Evolution and Education:** Anthropic didn't just release MCP and walk away; they have been actively improving it and educating developers. During the recent AI Summit, Anthropic's Mahesh Murthy delivered a workshop that went viral, accelerating MCP adoption. (Remember, all links for further learning are included at the end of the article.)



Why MCP Won (in short)

aka "won" status as de facto standard, over not-exactly-equivalent-but-alternative approaches like OpenAPI and LangChain/LangGraph. In rough descending order.

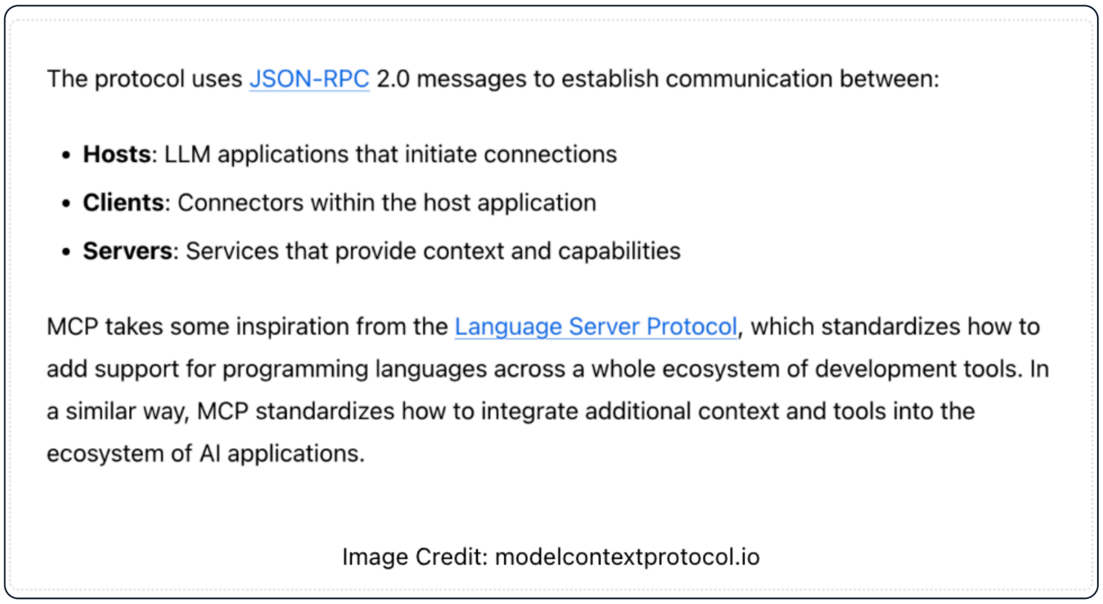
1. **MCP is "AI-Native" version of old idea**
2. **MCP is an "open standard" with a big backer**
3. **Anthropic has the best developer AI brand**
4. **MCP based off LSP, an existing successful protocol**
5. **MCP dogfooded with complete set of 1st party client, servers, tooling, SDKs**
6. **MCP started with minimal base, but with frequent roadmap updates**
7. **Non-Factors:** Things that we think surprisingly did *not* contribute to MCP's success
   - Lining up launch partners like Zed, SourceGraph Cody, and Replit
   - Launching with great documentation

The version from swyx (Latent Space)

So, What Is MCP and How Does It Work?

MCP lays out clear rules for how AI can find, connect to, and use external tools – whether it's querying a database or running a command. This lets models go beyond their training data, making them more flexible and aware of the world around them.

Technical Overview of MCP:



The protocol uses JSON-RPC 2.0 messages to establish communication between:

- **Hosts**: LLM applications that initiate connections
- **Clients**: Connectors within the host application
- **Servers**: Services that provide context and capabilities

MCP takes some inspiration from the Language Server Protocol, which standardizes how to add support for programming languages across a whole ecosystem of development tools. In a similar way, MCP standardizes how to integrate additional context and tools into the ecosystem of AI applications.

Image Credit: modelcontextprotocol.io

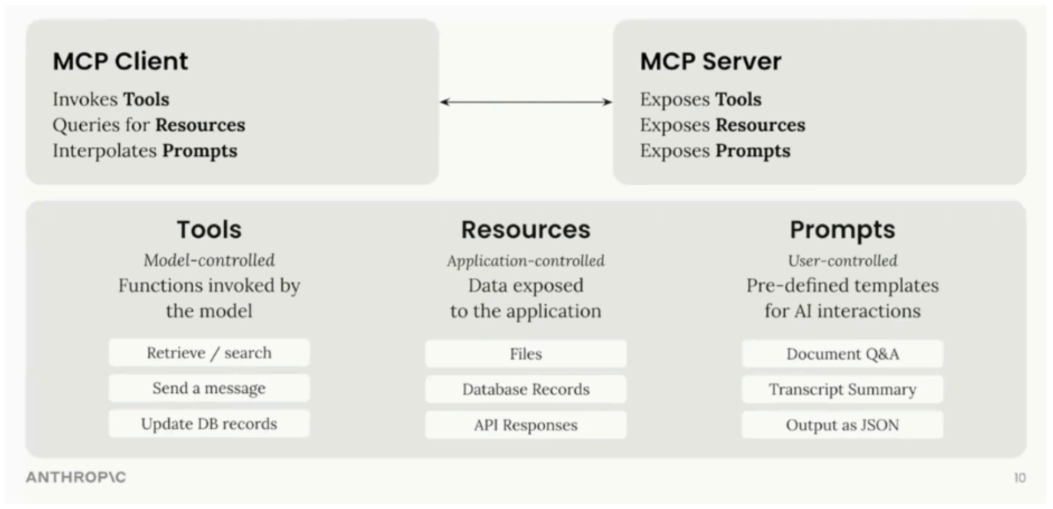


Image Credit: Anthropic

One striking feature is MCP's dynamic discovery – AI agents automatically detect available MCP servers and their capabilities, without hard-coded integrations. For example, if you spin up a new MCP server (like a CRM), agents can immediately recognize and use it via a standardized API, offering flexibility traditional approaches can't match.

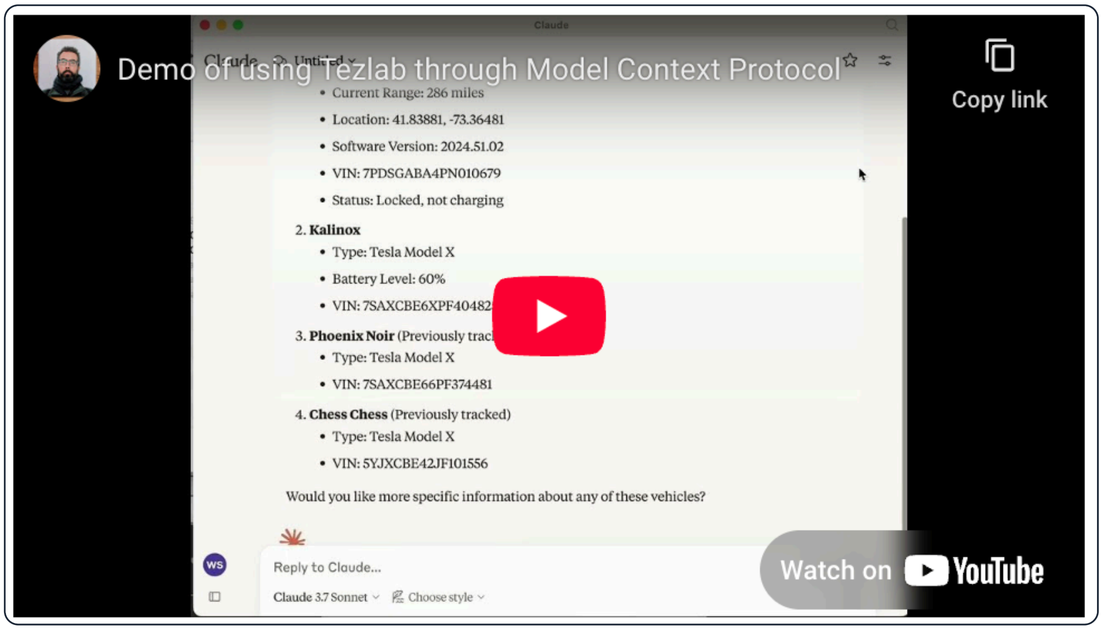
How do I actually get started with MCP?

The best place to start is the official MCP documentation and repository. Anthropic open-sourced the spec and provided SDKs (in languages like Python and now even Java). The steps typically are:

- **Run or install an MCP server for the tool or data source you care about**. Anthropic has an open-source repo of pre-built servers for popular systems (Google Drive, Slack, Git, databases, etc.). You can install these and configure them (often just running a command with your credentials or keys).
- **Set up the MCP client in your AI app.** If you're using Claude's app, you can add the server in the UI. If you're coding your own agent, use the MCP SDK to connect to the server (providing the address/port).
- **Once you've enabled the MCP services in your client, the client will pick on the additional functionality provided**: additional tools, resources and prompt templates.
- **Invoke and iterate.** The model/agent can now call the MCP tool actions as needed. Make sure to monitor logs to see that it's calling the servers correctly. You'll see requests hitting the MCP server and responses coming back.

For a quick start, Anthropic recommends trying the Claude Desktop integration (if you have access) or running the example servers and using their provided quickstart guide. The community is also very active – there is a rapidly expanding catalog of MCP servers. Some of the popular ones include connectors for Google services (Drive, Gmail, Calendar), Slack (chat and file access), GitHub/Git (for code repositories), databases like Postgres, web browsers or Puppeteer (to browse web pages), and many more. Many servers are listed in community directories (some developers have created sites to index them). The official MCP GitHub also hosts a bunch of connector implementations to get you started. And if you have a niche tool that isn't covered, you can build your own MCP server using the SDK – often it's just a thin wrapper around that tool's API, exposing a function in the MCP format.

We thank Will Schenk for clarifying a few things about MCP and how to start with it. He shared this quick hands-on walkthrough with Tezlab's Tesla monitoring service to *demonstrate MCP at work.*



- Current Range: 286 miles
- Location: 41.83881, -73.36481
- Software Version: 2024.51.02
- VIN: 7PDSGABA4PN010679
- Status: Locked, not charging

2. **Kalinox**
   - Type: Tesla Model X
   - Battery Level: 60%
   - VIN: 7SAXCBE6XPF40482
3. **Phoenix Noir** (Previously trac
   - Type: Tesla Model X
   - VIN: 7SAXCBE66PF374481
4. **Chess Chess** (Previously tracked)
   - Type: Tesla Model X
   - VIN: 5YJXCBE42JF101556

Would you like more specific information about any of these vehicles?

Before MCP, How Were AI Systems Handling Context And Tool Access?

Let's briefly look at the traditional approaches to giving AI external knowledge or actions, and how MCP differs:

- **Custom API Integrations (One-off Connectors):** The most common method has been writing custom code or using SDKs for each service. For example, if you wanted your AI agent to access Google Drive and a SQL database, you'd integrate Google's API and a database driver separately, each with its own authentication, data format, and quirks. Pain in the neck! MCP, by contrast, gives a single "key" (protocol) that can unlock many doors, and new MCP servers can be added without changing the client.
- **Language Model Plugins (OpenAI Plugins, etc.):** Another approach introduced in 2023 was providing the model a standardized plugin specification (often an OpenAPI schema) so it could call external APIs in a controlled way (e.g. the ChatGPT Plugins system). While conceptually similar to MCP (standardizing tool access), these were proprietary and limited – each plugin still needed to be built and hosted individually, and only certain platforms (like ChatGPT or Bing Chat) could use them. Plugins also tended to

focus on one-way data retrieval (the model calls an API and gets info) rather than maintaining an ongoing interactive session. MCP distinguishes itself by being open-source and universal (anyone can implement it, not tied to one AI provider) and by supporting rich two-way interactions. It's like a dialogue between the AI and tools, whereas plugins were often stateless question-answer calls.

- **Tool Use via Frameworks (LangChain tools, Agents):** Agent orchestration libraries like LangChain popularized the idea of giving models "tools" (functions) with descriptions. For example, you might have a search() tool or a calculate() tool, and the agent (via the LLM) decides when to invoke them. This is powerful, but each tool still required custom implementation under the hood – LangChain's library grew to 500+ tools implemented in a consistent interface, yet developers still had to wire up those tools or ensure they fit their needs. MCP can be seen as complementary here: it provides a standardized interface for the implementation of tools. In fact, you can think of MCP servers as a library of ready-made tools that any agent can use. The difference is where the standardization lies. LangChain created a developer-facing standard (its Tool class interface) to integrate tools into an agent's code. MCP creates a model-facing standard – the running AI agent itself can discover and use any MCP-defined tool at runtime. This means even if you don't custom-build an agent's code for a particular tool, the model can integrate it on the fly. In practice, these ideas are converging: for example, LangChain's team (when noticed the surge of MCP) provided an adapter so that all those MCP servers (connectors) can be treated as LangChain tools easily. So an agent built inLLangChain or other frameworks can call MCP tools just like any other, benefiting from the growing MCP ecosystem.
- **Retrieval-Augmented Generation (RAG) and Vector Databases:** A prevalent way to supply context to LLMs is to use a retriever that searches a knowledge base (documents, embeddings) and injects the top results into the prompt. This addresses the knowledge cutoff or limited memory of models. However, RAG usually deals with static text snippets and doesn't inherently let the model perform actions or queries beyond what's indexed. MCP can actually work alongside RAG – for instance, an MCP server could interface with a vector database or search engine, allowing the model to issue search queries as a tool rather than implicitly relying on retrieval every prompt. One could argue MCP is a more general mechanism: where RAG gives passive context, MCP lets the

model actively fetch or act on context through defined channels. In scenarios where up-to-date or interactive data is needed (say, querying a live database or posting an update), MCP extends beyond just retrieving text – it can trigger operations.

Is MCP a Silver Bullet and Solve-It-All?

Of course, MCP is not a silver bullet, it is an extremely convenient integration layer. But like any emerging technology, it introduces its own set of complexities and challenges that developers and organizations must consider before adopting it at scale: One of the primary concerns is the **added overhead of managing multiple tool servers**. Running and maintaining connections to these local servers can be cumbersome, particularly in production environments where uptime, security, and scalability are paramount. MCP's initial implementation was designed for local and desktop use, which raises questions about how well it translates to cloud-based architectures and multi-user scenarios. Developers have proposed making MCP more stateless and adaptable to distributed environments, but this remains an ongoing challenge. Another issue lies in **tool usability**. Just because MCP expands an AI model's toolset does not necessarily mean the model will use those tools effectively. Previous agent-based frameworks have demonstrated that AI models can struggle with tool selection and execution. MCP attempts to mitigate this by providing structured tool descriptions and specifications, but success still hinges on the quality of these descriptions and the AI's ability to interpret them correctly. The community-driven approach, as highlighted by LangChain's founder Harrison Chase, suggests that well-documented tools can enhance usability, but this is still an area of ongoing refinement. Beyond implementation hurdles, **MCP's maturity is also a consideration**. As a relatively new technology, it is subject to rapid changes and evolving standards. This can lead to breaking changes, requiring frequent updates to servers and clients. While the core concept of MCP appears stable, developers should anticipate and prepare for version upgrades and evolving best practices. **Compatibility is another limiting factor.** Currently, MCP has first-class support within Anthropic's ecosystem (e.g., Claude), but broader adoption remains uncertain. Other AI providers may not natively support MCP, requiring additional adapters or custom integrations. Until MCP gains wider acceptance across AI platforms, its utility will be somewhat constrained. For simpler applications, **MCP**

may even be overkill. If an AI model only needs to access one or two straightforward APIs, direct API calls might be a more efficient solution than implementing MCP. The learning curve associated with MCP's messaging system and server setup means that its benefits need to be weighed against its complexity. **Security and monitoring also present ongoing challenges.** Since MCP acts as an intermediary, it necessitates robust authentication and permission controls to prevent unauthorized access. Open-source initiatives like MCP Guardian have emerged to address these concerns by logging requests and enforcing policies, but securing MCP in enterprise environments remains a work in progress.

Overall, **none of these limitations are show-stoppers, but it's wise to start with experimental or non-critical deployments to get a feel for it.** One of the best things about MCP – the engaged community. Since it's open, issues you face can be discussed and addressed collaboratively.

MCP in Agentic Orchestration and Its Place in the Agentic Workflow

In previous articles, we explored the building blocks of autonomous agents: Profiling (identity and context), Knowledge, Memory, Reasoning/Planning, Reflection, and Action. An agent needs to observe and understand its environment (profile/knowledge), remember past interactions (memory), plan its moves (reasoning), take actions (execute tool calls or outputs), then reflect and learn. Where does MCP come in?

MCP is not itself an "agent framework"; rather, it acts as a standardized integration layer for agents. MCP is all about the Action part – specifically, giving agents a standardized way to perform actions involving external data or tools. It provides the plumbing that connects an AI agent to the outside world in a secure, structured manner. Without MCP (or something like it), every time an agent needs to do something in the world – whether fetching a file, querying a database, or invoking an API – developers would have to wire up a custom integration or use ad-hoc solutions. That's like building a robot but having to custom-craft each finger to grasp different objects – tedious and not scalable.

It’s important to highlight again that MCP is not an orchestration engine or agent brain by itself. Rather, it’s an integration layer within an agentic architecture. It complements agent orchestration tools like LangChain, LangGraph, CrewAI, or LlamaIndex by serving as a unified "toolbox" from which AI agents can invoke external actions. Instead of replacing orchestration – which determines when and why an agent uses a tool – MCP defines how these tools are called and information exchanged.

It is akin to a standardized API gateway for agents, reducing integration complexity from an "N×M" to an "N+M" problem by allowing universal compatibility between clients (agents) and servers (tools). Ultimately, MCP streamlines the integration of external functionalities, making agents more versatile, adaptable, and capable of performing sophisticated tasks across diverse contexts.

New Possibilities Unlocked by MCP

MCP is still new, and its full potential is just being explored. The first wave of use cases is obvious – connecting enterprise data to chat assistants or enhancing coding agents with repository access. But some emerging applications could take AI agents to the next level.

- **Multi-Step, Cross-System Workflows Agentic systems often need to coordinate across platforms.** Say an AI plans an event: it checks your calendar, books a venue, emails guests, arranges travel, and updates a budget sheet. Right now, this requires stitching APIs together manually. With MCP, all these actions happen through a single interface. The agent calls a series of MCP tools (one for each task), keeping shared context across them—no lost threads, no custom integrations.
- **Agents That Understand Their Environment (including Robotics)** Beyond tool access, MCP can enable AI agents embedded in smart environments – whether in a smart home or an operating system. An AI assistant could interact with sensors, IoT devices, or OS functions via standardized MCP servers. Instead of operating in isolation, the AI gains real-time awareness, enabling more natural and proactive assistance.

- **Collaborating Agents (Agent Societies)** – *I'm very excited about this one* – MCP could also serve as a shared workspace for multi-agent systems. Specialized AI agents – one for research, one for planning, another for execution – could use MCP to exchange information and coordinate tasks dynamically. With MCP, each agent doesn't need direct integrations; they simply access a common toolset.
- **Personal AI Assistants with Deep Integration MCP** could let users configure their own AI to interact with personal data and apps securely. A local MCP server could grant an AI access to emails, notes, and smart devices without exposing sensitive data to third parties. This could create an ultra-personalized AI assistant without relying on cloud-based services.
- **Enterprise Governance and Security For businesses**, MCP standardizes AI access to internal tools, reducing integration overhead. It also enables governance: AI interactions can be logged, monitored, and controlled via an oversight layer, preventing unintended actions while maintaining efficiency.

These are just the early glimpses of MCP's potential. By enabling fluid, context-aware, multi-step interactions, it moves AI agents closer to true autonomous workflow execution.

Concluding Thoughts

MCP is rapidly maturing into a powerful standard protocol that turns AI from an isolated "brain" into a versatile "doer." By streamlining how agents connect with external systems, it clears the path for more capable, interactive, and user-friendly AI workflows.

Key Upcoming Features (based on the workshop from Mahesh Murag from Anthropic)

Remote Servers & OAuth

- Seamless remote hosting using SSE.
- Built-in OAuth 2.0 for secure integration (e.g., Slack).

Official MCP Registry

- Centralized discovery and verification of servers.
- Enterprise-friendly: hosts can run private registries.

Well-Known Endpoints

- Standardized .well-known/mcp files for first-party server discovery.

Further Enhancements

- Streaming support, stateless connections, proactive server behavior, and better name spacing.

Each update will make MCP more robust, helping AI agents integrate more deeply into real-world workflows. **It's a community-driven effort, so keep an eye on the roadmap, join the discussions, and help shape the future of how AI and software intersect.**

MCP surged, and we even had to change our editorial schedule for it. This topic just begged to be explained. It felt only natural to cover it after discussing Action in agentic workflows. In the next episode, we will explore Human-AI communication and Human-in-the-Loop (HITL) integration, and then move on to Multi-Agent Collaboration. **Stay tuned.**

Sharing this article helps us grow and reach more people – thank you!

Resources to Dive Deeper:

- Introducing the Model Context Protocol by Anthropic
- Model Context Protocol documentation and quickstart guide
- MCP docs
- Model Context Protocol on GitHub
- Collection of Servers for MCP on GitHub
- Building Agents with Model Context Protocol (and especially the part: What's next for MCP) by Mahesh Murag from Anthropic, Workshop @AI Engineering Summit

- Why MCP Won by swyx from Latent Space
- GitHub Star History (charts)
- MCP: Flash in the Pan or Future Standard? by LangChain
- MCP Guardian on Github
- Exposing Services with MCP
- Initial reaction to MCP on reddit

Sources from Turing Post

- 🦸🏻 #1: Open-endedness and AI Agents – A Path from Generative to Creative AI?
- 🦸🏻 #5: Building Blocks of Agentic Systems
- 🦸🏻 #9: Does AI Remember? The Role of Memory in Agentic Workflows
- 🦸🏻 #10: Does Present-Day GenAI Actually Reason?
- 🦸🏻 #11: How Do Agents Plan and Reason?
- 🦸🏻 #12: How Do Agents Learn from Their Own Mistakes? The Role of Reflection in AI
- 🦸🏻 #13: Action! How AI Agents Execute Tasks with UI and API Tools

Thank you for reading!

📨 **If you want to receive our articles straight to your inbox, please** subscribe here

---

👏 Community

loaspra 4 days ago

I like to view MCP as a higher lever of abstraction on tool usage. First came langchain with its tool functions, and now we have MCP servers that encapsulate a full feature and

usability of a specific framework.
At first I thought that MCP were more related with the 'reasoning' part of the agent (well they maybe are bc the way tools are defined on the MCP server --markdown like instructions for each tool and the purpose of the MCP server-- they improve the overall performance of the agent).
But the most interesting fact about this is that Agent workflows will became more complex. Then we will improve even more the reasoning part (overall intelligence of the agents), maybe we will see more adversarial style of agents (just like MoE but on a higher level).

Interesting times ahead

1 reply · 🚀 1 +

Kseniase Article author about 21 hours ago

Totally agree!

+

Reply in thread

Mario1982 3 days ago

Thank you for this very interesting article!

1 reply · 🚀 1 +

Kseniase Article author about 21 hours ago

You are very welcome

+

Reply in thread

Edit Preview

Start discussing this article

Tap or paste here to upload images

Comment