AI DISCLOSURES PROJECT

SEPTEMBER 2025

POLICY BRIEF

MCP in Practice

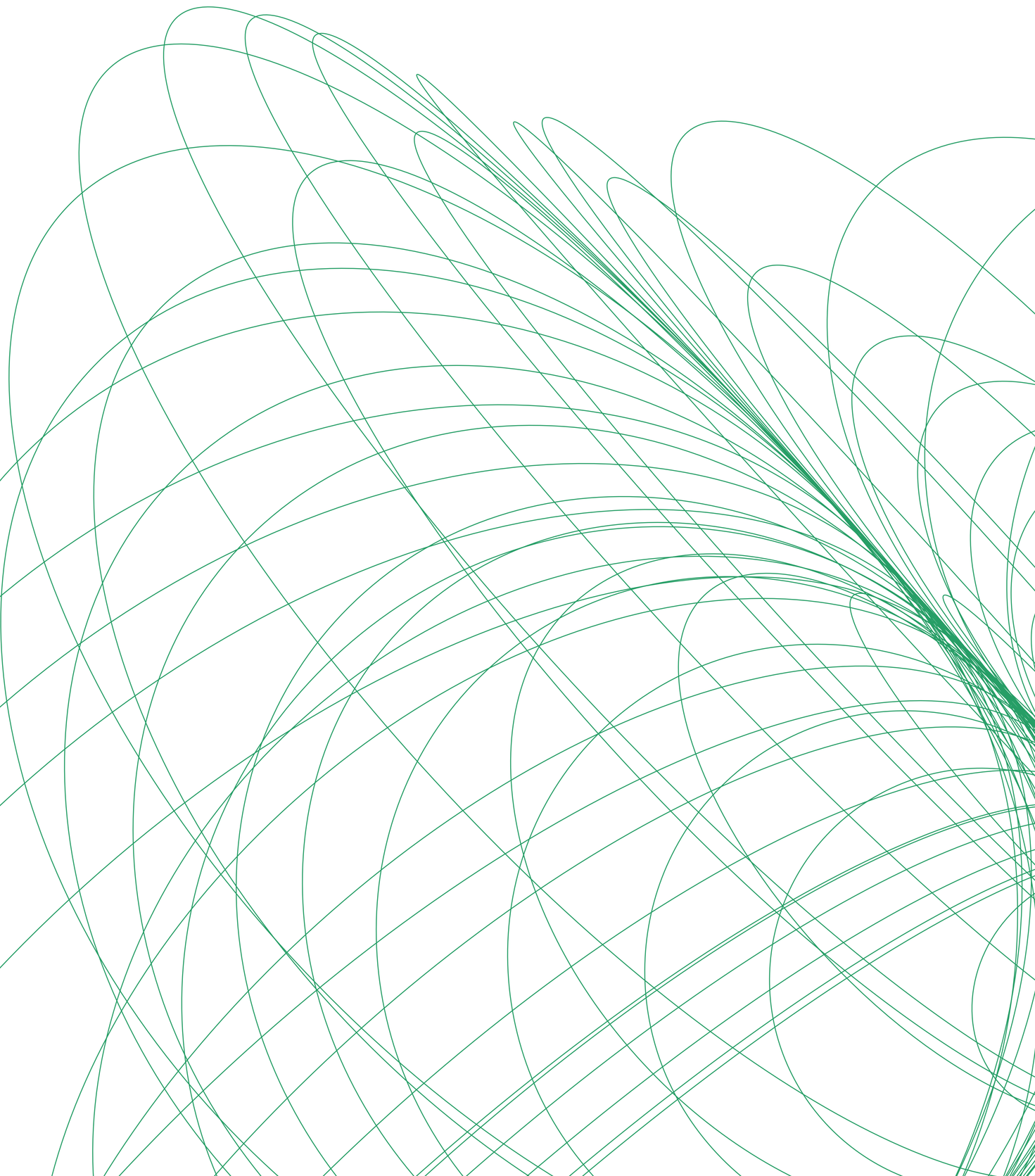
Mapping power, concentration, and usage in the emerging AI developer ecosystem

Sruly Rosenblat
Program Associate
Social Science Research Council

Ilan Strauss
Program Director
Social Science Research Council

Tim O'Reilly
Program Director
Social Science Research Council

Isobel Moure
Program Associate
Social Science Research Council



100 YEARS | SOCIAL SCIENCE RESEARCH COUNCIL

About The AI Disclosures Project

Led by technologist Tim O'Reilly and economist Ilan Strauss, the AI Disclosures Project addresses the potentially harmful societal impacts of AI's unrestrained commercialization. By improving corporate and technological transparency and disclosure mechanisms, it aims to ensure that economic incentives don't compromise safety or equity, and avoid fostering excessive risks. Disclosures are vital for well-functioning markets yet remain lacking in AI. Just as financial disclosure standards fostered robust securities markets, standardized AI disclosures can build trust, expedite adoption, and spur innovation. Through research, collaboration, and policy engagement, the AI Disclosures Project aims to develop a systematic framework for meaningful "Generally Accepted AI Management Principles." The project is generously funded by the Omidyar Network, Alfred P. Sloan Foundation, and Patrick J. McGovern Foundation.

DOI: 10.35650/AIDP.4119.d.2025

This policy brief can be referenced as follows:
Rosenblat, Sruly, Strauss, Ilan, O'Reilly, Tim, and Isobel Moure. "MCP in Practice." *Policy Brief.* Social Science Research Council, September 2025. https://www.ssrc.org/publications/mcp-in-practice-mapping-power-concentration-and-usage-in-the-emerging-ai-developer-ecosystem/

Originally published in Asimov Addendum, September 10, 2025:
https://asimovaddendum.substack.com/p/read-write-act-inside-the-mcp-server

100 YEARS | SOCIAL SCIENCE RESEARCH COUNCIL

Website: AI Disclosures Project
Substack: Asimov's Addendum
X: @AIDisclosures

1. The Rise And Rise Of MCP

Anthropic's Model Context Protocol (MCP) was released in November 2024 as a way to make tools and platforms model-agnostic. MCP works by defining servers and clients. MCP servers are local or remote endpoints where tools and resources are defined. For example, GitHub released an MCP server that allows LLMs to both read from and write to GitHub. MCP clients are the connection from an AI application to MCP servers — they allow an LLM to interact with context and tools from different servers. An example of an MCP client is Claude Desktop, which allows the Claude models to interact with thousands of MCP servers.

In a relatively short time, MCP has become the backbone of hundreds of AI pipelines and applications. Major players like Anthropic and OpenAI have built it into their products. Developer tools such as Cursor (a coding-focused text editor or IDE) and productivity apps like Raycast also use MCP. Additionally, thousands of developers use it to integrate AI models and access external tools and data without having to build an entire ecosystem from scratch.

In previous work published with AI Frontiers, we argued that MCP can act as a great unbundler of "context" – the data that helps AI applications provide more relevant answers to consumers. In doing so, it can help decentralize AI markets. **We argued that, for MCP to truly achieve its goals, it requires support from:**

1. **Open APIs:** so that MCP applications can access third-party tools for agentic use (write actions) and context (read).

2. **Fluid memory:** Interoperable LLM memory standards, accessed via MCP-like open protocols, so that the memory context accrued at OpenAI and other leading developers does not get stuck there, preventing downstream innovation.

Protocols are fundamentally market-shaping devices, architecting markets through the permissions, rules, and interoperability of the network itself.

We expand upon these two points in a recent policy brief for those looking to dig deeper.

More generally, **we argue that protocols, like MCP, are actually foundational "rules of the road" for AI markets**, whereby open disclosure and communication standards are built into the network itself, rather than imposed after the fact by regulators. Protocols are fundamentally market-shaping devices, architecting markets through the permissions, rules, and interoperability of the network itself. They can have a big impact on how the commercial markets built on top of them function too.

1.1 But How Is the MCP Ecosystem Evolving?

Yet we don't have a clear idea of the shape of the MCP ecosystem today. *What are the most*

common use cases of MCP? What sort of access is being given by MCP servers and used by MCP clients? Is the data accessed via MCP "read-only" for context, or does it allow agents to "write" and interact with it — for example, by editing files or sending emails?

To begin answering these questions, we look at the tools and context which AI agents use via MCP servers. This gives us a clue about what is being built and what is getting attention. In this article, we don't analyze *MCP clients* — the applications that use MCP servers. We instead limit our analysis to what MCP servers are making available for building.

In practice, *while there were plenty of MCP servers, we found that the top few garnered most of the attention and, likely by extension, most of the use*. **Just the top 10 servers had nearly half of all GitHub stars given to MCP servers.**

Some of our takeaways are:

1. *MCP usage appears to be fairly concentrated.* This means that, if left unchecked, a small number of servers and (by extension) APIs could have outsized control over the MCP ecosystem being created.

2. *MCP use (tools and data being accessed) is dominated by just three categories:* Database & Search (RAG), Computer and Web Automation, and Software Engineering. Together, they received nearly three-quarters (72.6%) of all 'stars' on GitHub (which we proxy for usage).

3. Most MCP servers support both *read* (access context) and *write* (changing context) operations, showing that developers want their agents to be able to act on context, not just consume it.

2. Data Collection

We assembled a large dataset of MCP servers (n = 2,874), scraped from Pulse MCP. We then enriched it with GitHub star count data on each server. On GitHub, stars are similar to Facebook "likes," and developers use them to bookmark projects, show appreciation for the project developer or indicate usage.

We categorized each repo into one of 15 categories using the GPT-5 Mini model. We then human-reviewed and edited the top 50 servers that make up around 70% of the total star count in our dataset.

The full dataset, along with descriptions of the categories, can be found here: https://huggingface.co/datasets/sruly/MCP-In-Practice

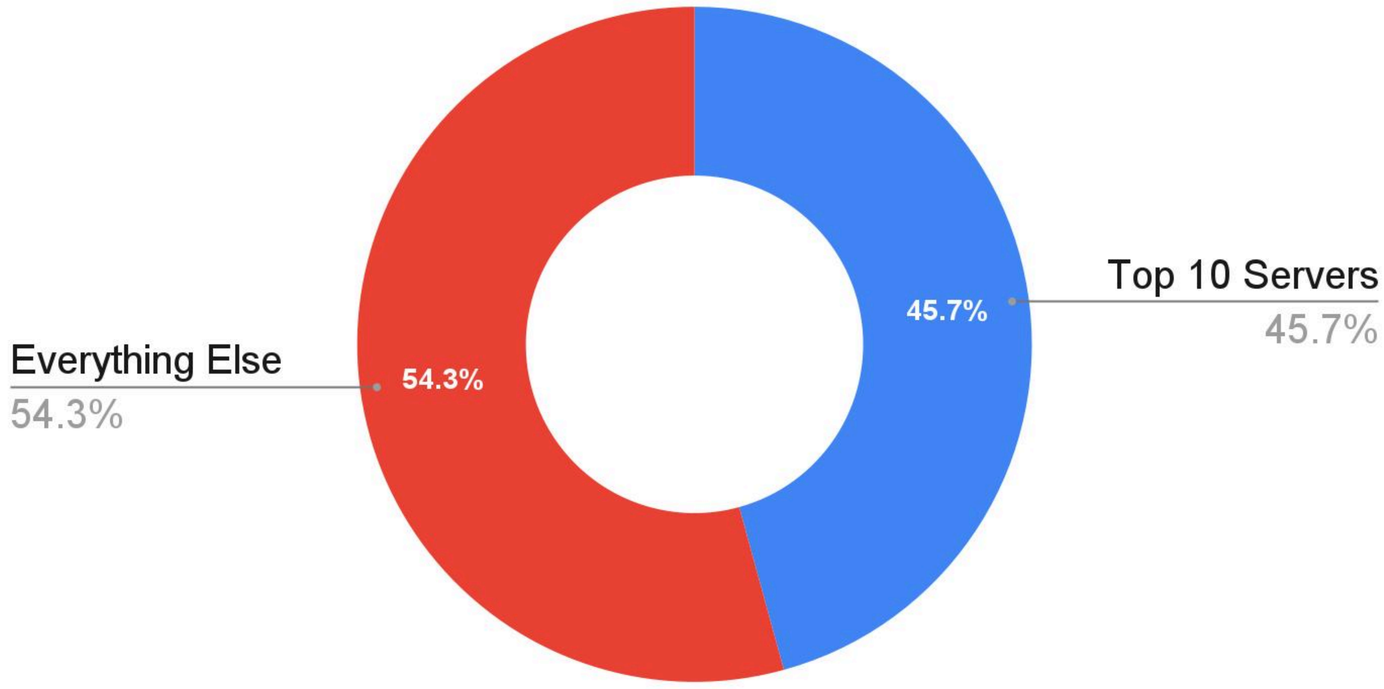
3. Findings

To start, we analyzed the MCP ecosystem for concentration risk.

3.1 MCP Server Use Is Concentrated

We found that MCP usage is concentrated among several key MCP Servers, judged by the number of GitHub stars each repo received.

Despite there being thousands of MCP servers, the top 10 servers make up nearly half (45.7%) of all GitHub stars given to MCP servers (pie chart below) — and the top 10% of servers make up 88.3% of all GitHub stars (not shown).



The top 10 servers received 45.7% of all GitHub stars in our dataset of 2,874 servers.

This means that the majority of real-world MCP users are likely relying on the same few services made available via a handful of APIs. This concentration likely stems from network effects and practical utility: all developers gravitate toward servers that solve universal problems like web browsing, database access, and integration with widely used platforms like GitHub, Figma, and Blender. This concentration pattern seems typical of developer-tool ecosystems. A few well-executed, broadly applicable solutions tend to dominate. Meanwhile, more specialized tools occupy smaller niches.

3.2 The Top 10 MCP Servers Really Matter

The top 10 MCP servers are shown in the following table, along with their star count and what they do.

Among the top 10 MCP servers, *GitHub, Repomix, Context7,* and *Framelink* are built to assist with software development: *Context7* and *Repomix* by gathering context, *GitHub* by allowing agents to interact with projects, and *Framelink* by passing on the design specifications from Figma directly to the model. The *Blender* server allows agents to create 3D models of anything, using the popular open-source *Blender* application. Finally, *Active Pieces* and *MindsDB* connect the agent to multiple APIs with one standardized interface; in *MindsDB*'s case, primarily to read data from databases, and Active Pieces to automate services.

MCP SERVER	GITHUB STARS	WHAT IT DOES
Browser Use	61,000 ⭐	Controls your web browser to automatically visit sites, extract data, and fill forms
Minds DB	35,433 ⭐	Connects AI to third party databases so you can ask questions about your data
Context7 MCP	27,829 ⭐	Provides up-to-date code documentation for LLMs and AI code editors
Github MCP	21,885 ⭐	Lets AI read, search, and manage code repositories and software projects
TaskMaster	21,223 ⭐	An AI-powered task management system
Repomix	18,813 ⭐	Packages and compresses code projects for use by LLM models
Playwright MCP	18,425 ⭐	Lets AI control a headless browser (like Chrome) for testing and automating web browsing
ActivePieces	16,495 ⭐	Connects different apps together to create automated workflows without coding
Blender MCP	13,035 ⭐	Controls Blender 3D software to create and edit 3D models and animations
Framelink Figma MCP	10,296 ⭐	Access Figma design files to help guide coding

The top 10 MCP servers with short descriptions, design courtesy of Claude.

The dominance of agentic browsing, in the form of Browser Use (61,000 stars) and Playwright MCP (18,425 stars), stands out. This reflects the fundamental need for AI systems to interact with web content. These tools allow AI to navigate websites, click buttons, fill out forms, and extract data just like a human would. *Agentic browsing has surged even though it's far less token-efficient than calling an API.* Browsing agents often need to wade through multiple pages of boilerplate to extract slivers of data a single API request could return. Because many services lack usable APIs or tightly gate them, browser-based agents are often the simplest—sometimes the only—way to integrate, underscoring the limits of today's APIs.

Some of the top servers are unofficial. Both the *Framelink* and *Blender MCP* are servers that interact with just a single application, but they are both "unofficial" products. This means that they are not officially endorsed by the developers of the application they are integrating with – those who own the underlying service or API (e.g., GitHub, Slack, Google). Instead, they are built by independent developers who create a bridge between an AI client and a service — often by reverse-engineering APIs, wrapping unofficial SDKs, or using browser automation to mimic user interactions.

It is healthy that third-party developers can build their own MCP servers, since this openness encourages innovation. But it also introduces an intermediary layer between the user and the API, which brings risks around trust, verification, and even potential abuse. With open-source local servers, the code is transparent and can be vetted. By contrast, remote third-party servers are harder to audit, since users must trust code they can't easily inspect.

At a deeper level, the repos that currently dominate MCP servers highlight three encouraging facts about the MCP ecosystem:

1. **First, several prominent MCP servers support multiple third-party services for their functionality.** *MindsDB* and *Active Pieces* serve as gateways to multiple (often competing) service providers through a single server. MindsDB allows developers to query different databases like PostgreSQL, MongoDB, and MySQL through a single interface, while Taskmaster allows the agent to delegate tasks to a range of AI models from OpenAI, Anthropic, and Google, all without changing servers.

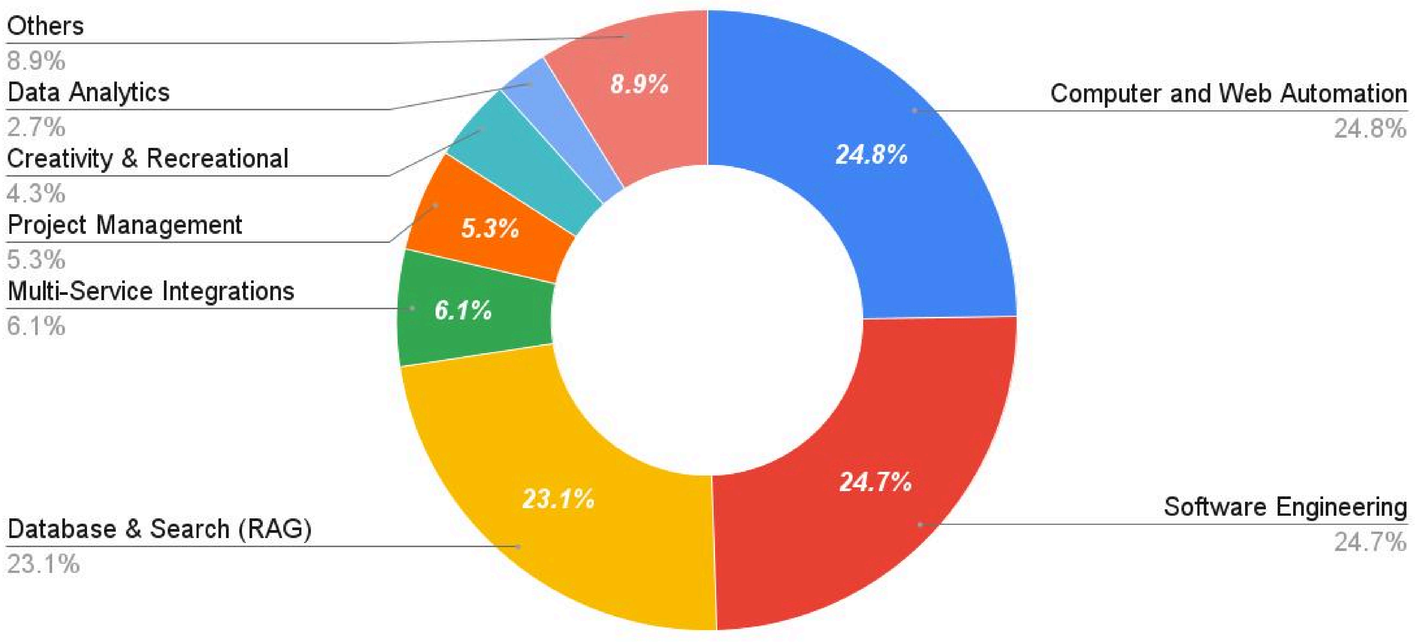
2. **Second, agentic browsing MCP servers are being used to get around potentially restrictive APIs.** As noted above, Browser Use and Playwright access internet services through a web browser, helping to bypass API restrictions, but they instead run up against anti-bot protections. This circumvents the limitations that APIs can impose on what developers are able to build.

3. **Third, some MCP servers do their processing on the developer's computer (locally), making them less dependent on a vendor maintaining API access.** *Some MCP servers examined here can run entirely on a local computer without sending data to the cloud, meaning no gatekeeper has the power to cut you off.* Of the 10 MCP servers examined above, only *Framelink*, *Context7*, and *GitHub* rely on just a single cloud-only API dependency that can't be run locally end-to-end on your machine. *Blender* and *Repomix* are completely open-source and don't require any internet access for them to work, while *MindsDB*, *Browser Use*, and *Active Pieces* have local, open-source implementations.

3.3 The Three Categories That Dominate MCP Use

Next, we grouped MCP servers into different categories based on their functionality.

When we analyzed what types of servers are most popular, we found that three dominated: **Computer & Web Automation (24.8%), Software Engineering (24.7%), and Database & Search (23.1%).**



Software Engineering, Computer & Web Automation, and Database & Search received 72.6% of all stars given to MCP servers.

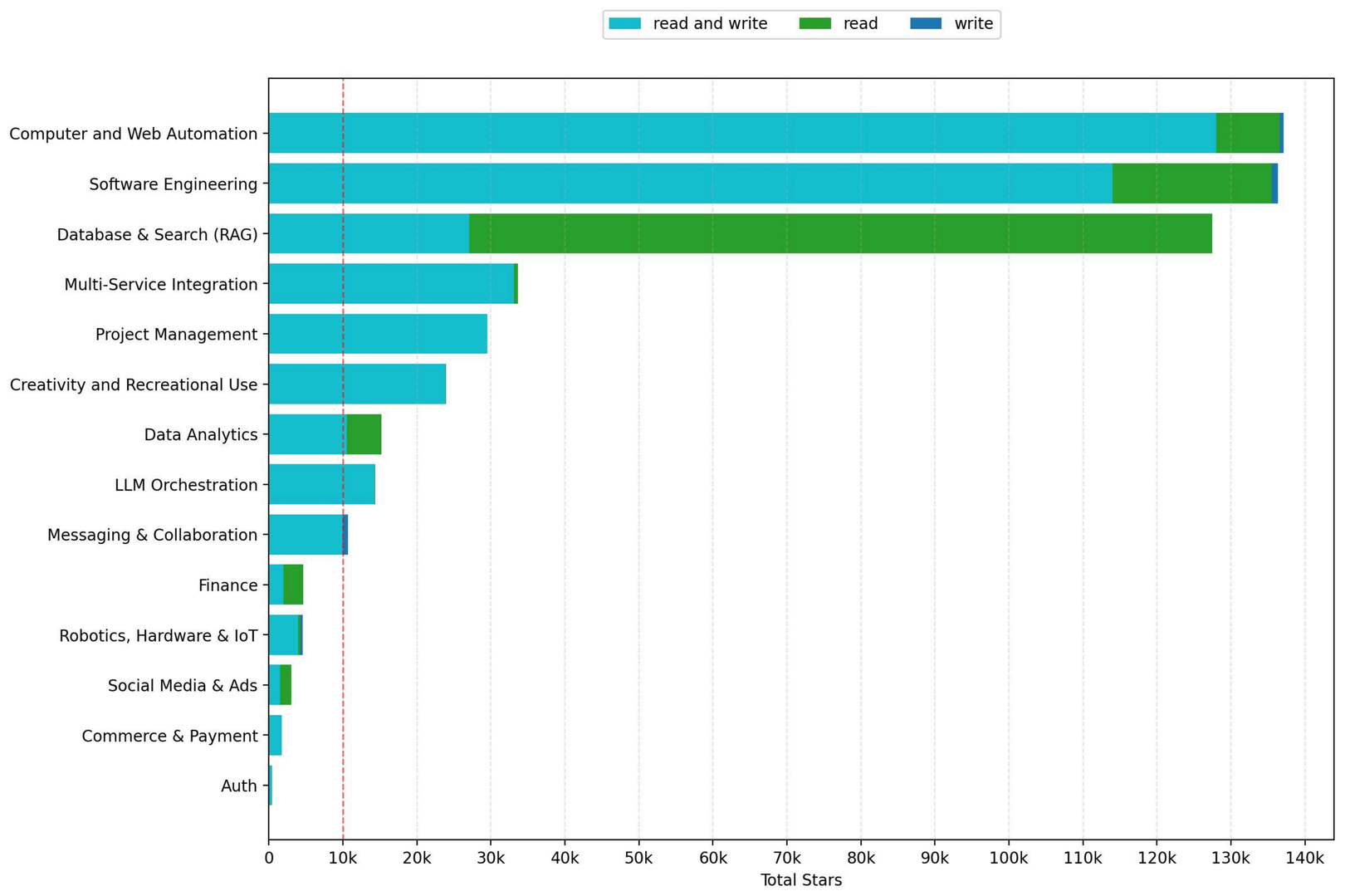
Widespread use of Software Engineering (24.7%) MCP servers aligns with Anthropic's economic index, which found that an outsized portion of AI interactions was software development related.

The popularity of both Computer & Web Automation (24.8%) and Database & Search (23.1%) also makes sense. Before the advent of MCP, web scraping and database search were highly integrated applications across platforms like ChatGPT, Perplexity, and Gemini. With MCP, however, users can now access that same search functionality and connect their agents to any database with minimal effort. In other words, MCP's unbundling effect is highly visible here.

3.4 Agents Interact With Their Environments

Lastly, we analyzed the capabilities of these servers: are they allowing AI applications just to access data and tools (*read*), or instead do agentic operations with it (*write*)?

Across all but two of the MCP server categories looked at, the most popular MCP servers supported both *reading* (access context) and *writing* (agentic) operations—shown in turquoise. The prevalence of servers with combined read and write access suggests that agents are not being built just to answer questions based on data, but also to take action and interact with services on a user's behalf.



Showing MCP servers by category. Dotted red line at 10,000 stars (likes). The most popular servers support both read and write operations by agents. In contrast, almost no servers support just write operations.

The two exceptions are Database & Search (RAG) and Finance MCP servers, in which *read-only* access is a common permission given. This is likely because data integrity is critical to ensuring reliability.

4. The Importance Of Multiple Access Points

A few implications of our analysis can be drawn out at this preliminary stage.

First, concentrated MCP server use compounds the risks of API access being restricted. As we discussed in Protocols and Power, MCP remains constrained by "*what a particular service (such as GitHub or Slack) happens to expose through its API*." A few powerful digital service providers have the power to shut down access to their servers.

One important hedge against API gatekeeping is that many of the top servers try not to rely on a single provider. **In addition, the following two safeguards are relevant**:

- **Offer local processing** of data on a user's machine whenever possible, instead of sending the data for processing to a third-party server. Local processing ensures that functionality cannot be restricted.

- If running a service locally is not possible (e.g., email or web search), the server should still **support multiple avenues of getting at the needed context through competing APIs**. For example, *MindsDB* functions as a gateway to multiple data sources, so instead of relying on just one database to read and write data, it goes to great lengths to support multiple databases in one unified interface, essentially making the backend tools interchangeable.

Second, our analysis points to the fact that current restrictive API access policies are not sustainable. Web scraping and bots, accessed via MCP servers, are probably being used (at least in part) to circumvent overly restrictive API access, complicating the increasingly common practice of banning bots. Even OpenAI is coloring outside the API lines, using a third-party service to access Google Search's results through web scraping, thereby circumventing its restrictive API.

Expanding structured API access in a meaningful way is vital. *This ensures legitimate AI automation runs through stable, documented endpoints*. Otherwise, developers resort to brittle browser automation where privacy and authorization have not been properly addressed. Regulatory guidance could push the market in this direction, as with open banking in the U.S.

Finally, encouraging greater transparency and disclosure could help identify where the bottlenecks in the MCP ecosystem are.

- Developers operating popular MCP servers (above a certain usage threshold) or providing APIs used by top servers should report usage statistics, access denials, and rate-limiting policies. This data would help regulators identify emerging bottlenecks before they become entrenched. *GitHub might facilitate this by encouraging these disclosures, for example.*

- Additionally, MCP servers above certain usage thresholds should clearly list their dependencies on external APIs and what fallback options exist if the primary APIs become unavailable. This is not only helpful in determining the market structure, but also essential information for security and robustness for downstream applications.

The goal is not to eliminate all concentration in the network, but to ensure that the MCP ecosystem remains contestable, with multiple viable paths for innovation and user choice. By addressing both technical architecture and market dynamics, these suggested tweaks could help MCP achieve its potential as a democratizing force in AI development, rather than merely shifting bottlenecks from one layer to another.

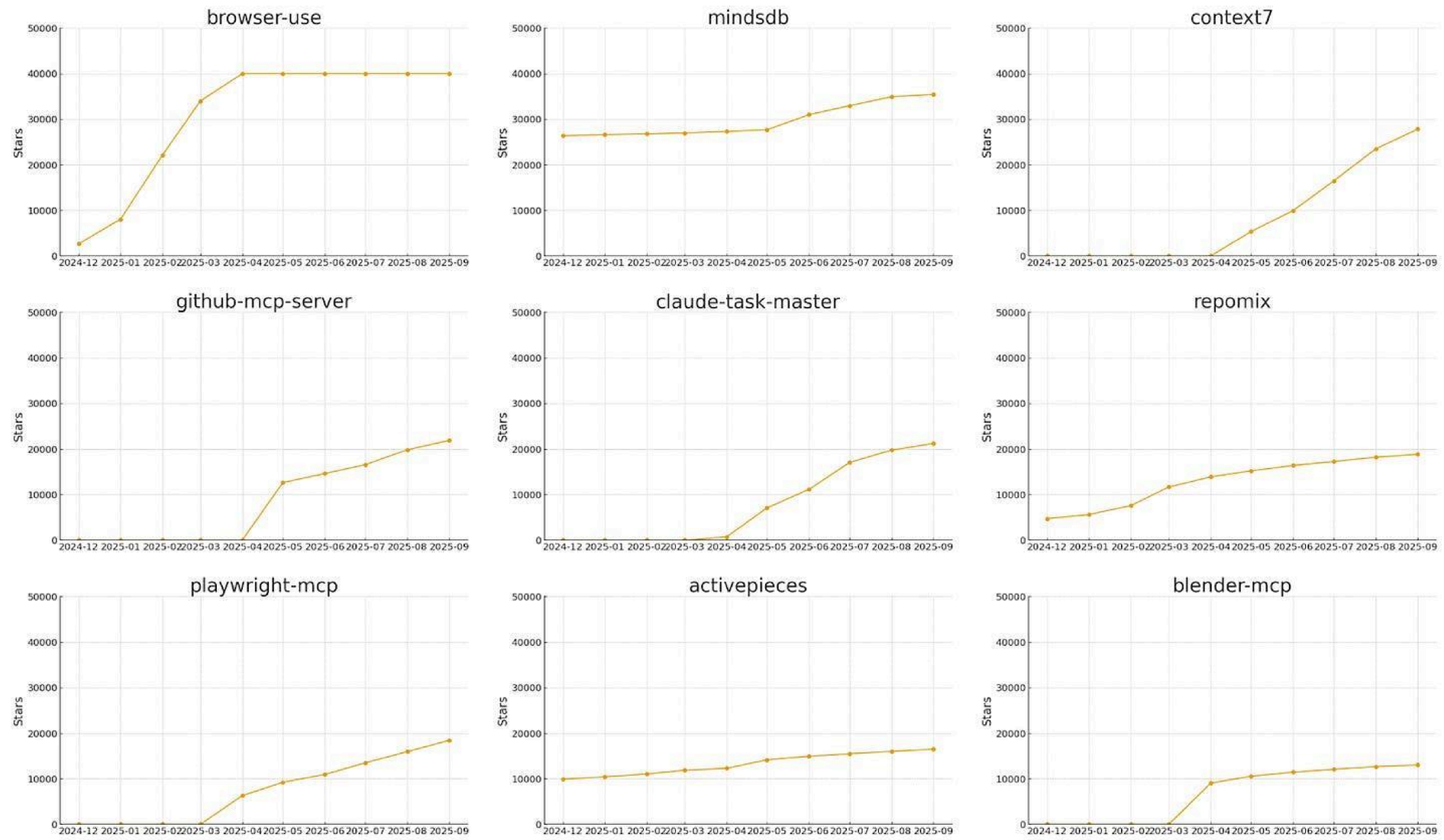
Appendix

Limitations:

There are a few limitations to our preliminary research:

- GitHub stars aren’t a measure of download counts or even necessarily a repo’s popularity.
- Only the name and description were used when categorizing repos with the LLM.
- Categorization was subject to both human and AI errors and many servers would likely fit into multiple categories.
- We only used the PulseMCP list for our dataset, other lists had different servers (e.g. Browser Use isn’t on mcpmarket.com).
- We excluded some repos from our analysis, such as those that had multiple servers and those we weren't able to fetch the star count for. We may miss some popular servers by doing this.

MCP Server Use Over Time



The growth of the top nine repos' star count over time from MCP's launch date on November 25th, 2024 until September 2025. NOTE: We were only able to track the browser-use's repo until 40,000 stars, hence the flat line for its graph. In reality, roughly 21,000 stars were added over the next few months (the other graphs in this blog are properly adjusted).