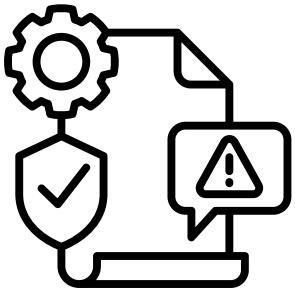


# MEMO: MODEL CONTEXT PROTOCOL (MCP)

## UNDERSTANDING MCP, AND FAD OR REAL?

**Model Context Protocol (MCP)** has been the rage in the AI-world over the last few months.

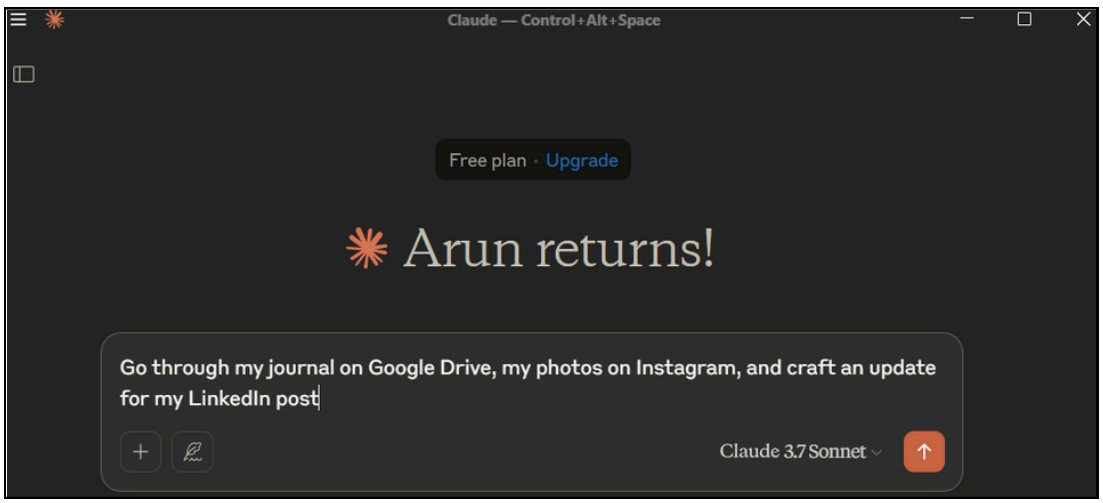
Before we critique this new technology, it is important to understand what MCP is, and how it works.

### Understanding MCP

Today, most of us use AI-tools, especially advanced language models, like ChatGPT, Gemini, or Grok by going to their website or using the respective app on the phone. However, there is one other way to use advanced language models: as *installed desktop tool on Windows or Mac* (this is possible with Claude Desktop and with Microsoft CoPilot).

Now, imagine you are on your Windows PC, using the Claude Desktop app (incidentally, MCP was proposed by Anthropic, the parent company of Claude).

Let me take a consumer use-case first, and this is what you type into Claude desktop: "Go through my journal on Google Drive, my photos on Instagram, and craft an update for my LinkedIn post" (*cringe!*). A few months ago, this would not have been possible, since AI-tools (like Claude or CoPilot) didn't have access to *tool use* – that is, they could not read your Google Drive, look at your Instagram, or access your LinkedIn.



This is what MCP makes possible: for your AI-tool to access your Google Drive, Instagram, and LinkedIn – in a safe & secure way – and then complete the action that you want.

The way this is done is for your AI-tool to use installed "MCP servers", which standardize the way in which AI-tools connect to 3rd party servers and services. To make it more clear, in the above case, the user would have to install a Google Drive MCP server, an Instagram MCP server, and a LinkedIn MCP server on their Windows computer, and then Claude Desktop would be able to complete the request (after authenticating each tool in turn). We will return to this later, but this is a *painful user experience*: installing multiple MCP servers, authenticating multiple times, and having to jump hoops to complete something natural.

So, why has MCP caught the attention of various AI-builders? An enterprise use-case makes it clearer. Let us say your IT-admin has mandated use of Github CoPilot Enterprise. Now, inside this secure & locked-down tool, you enter the following query: "Take the top 5 issues from our ServiceNow tickets, get the customer details from our Salesforce CRM for these issues, and create a Notion page using the customer-response template using the ticket summary and customer details".

This use-case suddenly pops! It is something that IT wants to mandate (for secure access to company data), while enabling use of the latest AI-tools. Also, tools like ServiceNow, Salesforce, and Notion are typically used in business scenarios, where integration across tools today is a pain.

This is where MCP really shines, both as a protocol and an end-user experience. As a protocol, it makes it possible for AI-tools (like Claude Desktop and CoPilot) to access and use a variety of 3rd party enterprise systems consistently and without extra coding. And, as an end-user experience, it allows for an end-user in a business context to direct AI-tools using natural, human language to accomplish something that cuts across multiple systems.

And lastly, for freelance developers & budding AI-developers inside companies – building MCP servers that connect to existing 3rd party systems is easy to do and uses existing knowledge: basically, an MCP server is typically a set of AI-discoverable interfaces that act as a passthrough to underlying enterprise system APIs. To put it simply and without technical jargon: MCP gives AI-tools a consistent way to find and use varied enterprise system capabilities.

The icing on the cake is the naming & locked down security access – model-context protocol – sounds sufficiently enterprise-y and to receive approval from CIOs...

## A Critique of MCP, and why I think it's a fad

The tragedy of someone who has been around technology for a while is that you've seen it all, and heard the same things go around and come back again in a different form.

In a weird way MCP reminds me of CORBA (common object resource broker architecture), a Java-fad and protocol from the late '90s and '00s. CORBA was a way for enterprise systems to consistently discover and use 3rd party code that was available on an object bus!

However, CORBA was a fad that died with the rise of SaaS, REST APIs, and micro-services architectures.

To put it simply, CORBA was backwards looking (*think "car is a horseless carriage"*) and assumed that enterprise systems would remain as they were, but would want to integrate with the new thing called the Internet by securely accessing code "blobs" that would complete certain tasks.

I feel the same way about Model Context Protocol: it assumes that enterprise systems are static and won't integrate AI themselves and move to new architectures, that the only way to deal with security is repeated access control requests, and that people inside the enterprise will continue to be happy with a sub-par experience to the consumer-grade experience.

When I imagine the future – even just a year out – it feels different: agents that act on our behalf, that are able to discover and use 3rd party (including enterprise) applications and services, that are able to authenticate us, and that are able to mediate all this with our security & privacy as a key goal.

So, the ideal user experience I can imagine in our near future is more like this: my personal Appa Agent has monitored my chat with my wife, knows that we have dinner plans for today, has made a reservation at a restaurant for the evening after checking with her agent, has monitored traffic conditions, and based on the drive time has made an Uber booking on my behalf. This requires interaction with multiple 3rd party systems & agents, along with authentication and so on, but it is made seamless through consent-based systems that will emerge to prioritize ease of user experience over stringent security... this is what the history of technological progress has shown us repeatedly.

## What then of MCP?

I think MCP is a fad, especially as an eventual end-user experience. However, it is not all a wash for MCP; infrastructure like MCP pushes the envelope, and while not ideal, makes it possible to light-up use cases, that then are improved by newer technologies. What then happens to the "stand-in" technologies like MCP? They either become key hidden building blocks of what replaces them, or they are relegated to curios experiments of the past – much like my fond memories of CORBA!

*Arun Rajappa*

@appa (twitter), LinkedIn
v0.1, 2025-04-27