Model Context Protocol

A Comprehensive Overview

By: Ashu Mishra
Technical Product Manager
LinkedIn Profile

Executive Summary

The Model Context Protocol (MCP) represents a paradigm shift in how AI systems interact with data sources and tools. This document synthesizes insights from a comprehensive presentation covering the evolution of AI adoption, the challenges of fragmented AI ecosystems, and how MCP provides a unified solution for context management across different platforms and tools.

Part 1: The Evolution of AI Adoption and The Problem of Silos

The Three Stages of AI Adoption

The integration of Large Language Models into everyday workflows has progressed through three transformative stages:

Stage 1: Experimentation (The Wonder Phase)

When ChatGPT launched, it achieved unprecedented growth: **1 million users in 5 days** and **100 million users in 2 months**. For the first time in software history, humans could interact with software without coding, treating it like everyday conversations. Early adopters experimented with creative prompts such as "Write a poem on Pizza in the style of Gulzar" or "What would happen if AI starts breeding tomorrow?" Social media platforms exploded with AI-generated content, showcasing the technology's creative potential.

Stage 2: Professional Adoption

As the novelty wore off, professionals began incorporating AI into their daily workflows. ChatGPT became an indispensable tool for:

- Text summarization and content generation
- Email composition and communication
- Lesson planning and educational content
- Code debugging and development assistance

This stage marked a significant increase in individual productivity, as AI transitioned from entertainment to practical utility.

Stage 3: The Ecosystem Explosion (The Now)

The current stage is characterized by proliferation of specialized AI tools and integrations:

- Microsoft Copilot became omnipresent across the Microsoft ecosystem
- Google Suite introduced comprehensive AI integrations
- Specialized tools emerged: Perplexity for research, Replit and Cursor for development
- Every major platform developed its own AI wrapper and integration

The Critical Problem: Fragmented AI Ecosystems

Despite this explosion of AI capabilities, a fundamental problem emerged: **the problem of silos**. Each AI tool operated within its own isolated ecosystem:

- AI in Notion couldn't communicate with AI in Slack

- Cursor couldn't access messages on Microsoft Teams
- Each AI agent had its own character and limitations
- Users possessed an "army of AI agents" but couldn't coordinate them effectively

Users found themselves juggling multiple AI assistants, each locked within its platform. The dream was clear: **one unified assistant that could understand their entire work context and solve problems across all platforms**. However, the biggest bottleneck preventing this unified AI experience was **context**.

Understanding Context: The Foundation of AI Intelligence

Context is everything an AI can see while generating a response. It encompasses:

- Conversation history
- External documents and data sources
- User preferences and past interactions
- Real-time information from connected tools

For example, when chatting with ChatGPT, the context includes all previous messages in the conversation. Without proper context, AI responses become generic and unhelpful. The challenge was enabling AI to access context from multiple sources simultaneously.

Part 2: The Context Problem and Evolution of Solutions

A Product Manager's Journey: Before and After AI

To illustrate the context problem, consider a typical day in the life of a Product Manager:

Traditional Workflow:

1. Discovery (researching user needs and market trends)
2. Analysis (evaluating data and insights)
3. Documentation (creating specs and requirements)
4. Request raising/development/Sprint planning
5. User Acceptance Testing

With ChatGPT Introduction: ChatGPT began assisting with analysis, documentation, and sprint planning. However, this created a new problem: the "Ctrl+C, Ctrl+V fallacy."

The Ctrl+C, Ctrl+V Fallacy

While AI could help with individual tasks, humans became "Human APIs" - constantly transferring data between tools to build context:

- Copying data from Slack to feed into ChatGPT
- Manually extracting information from Google Drive to provide context
- Transferring analysis results between different AI tools

Key problems emerged:

- Extensive data migration required between tools
- Context building time exceeded actual productive work time
- Non-sustainable workflow in the long run
- Fragmented knowledge across multiple platforms

The Solution: Function Calling and Tools

Function Calling Revolution

In late 2023, OpenAI introduced function calling, a breakthrough that enabled LLMs to interact with external systems. Function calling allows:

- LLMs to call extension files and external APIs
- Programmatic retrieval of structured data from GPT models

- JSON-based responses with function names and arguments

According to HackWithGPT, "OpenAI isn't executing the function for you, but it returns a JSON object with the function name it thinks should be called and the arguments to pass to the function."

The Advent of Tools

Function calling enabled a new generation of AI tools with specialized integrations:

- **Salesforce integration** for sales teams to access CRM data
- **Slack bots** reading chat history and channel context
- **Google Drive connectors** for document access and collaboration
- **Database query tools** analyzing organizational databases
- **GitHub integrations** for code review and pull request management
- **HR tools** for employee data access
- **Accounting system integrations** for financial management
- **Marketing tools** for campaign management
- **File system access** with intelligent code search
- **Web browsing** for real-time information retrieval

Major platforms rushed to add capabilities: ChatGPT introduced browsing, file uploads, and code execution; Claude gained computer access and tool integration; and parallel developments occurred across all major AI platforms.

The Integration Nightmare

While tools solved the context problem, they created a new challenge: the **N×M integration problem**.

The Mathematics of Integration Complexity:

- N clients (AI applications) × M servers (data sources) = N×M unique integrations required
- Each integration had different authentication methods
- Different data formats and APIs across platforms
- Inconsistent error handling and protocols

Consequences:

- Development nightmare with exponentially increasing complexity
- Maintenance burden across numerous custom integrations
- Security vulnerabilities from inconsistent authentication
- Significant cost and time wastage
- Scaling impossibility

Part 3: Model Context Protocol - The Universal Solution

The MCP Paradigm

Let's recap the problem: Every AI tool was creating its own API rules to integrate with platforms like GitHub. The solution was elegant: **GitHub builds an integration with standardized protocols that can be used by any AI tool**.

What is Model Context Protocol?

Model Context Protocol is a universal standard that defines how AI applications interact with data sources. Breaking down the name:

- **Model:** Any LLM (GPT, Claude, Gemini, etc.)
- **Context:** All information provided to the LLM
- **Protocol:** Standardized rules that everyone must follow

MCP eliminates the N×M problem by providing a single, standardized way for any AI application to connect with any data source.

MCP Architecture: The Phone Analogy

Think of MCP like a phone with multiple SIM card slots:

- **Phone with multiple SIM slots** = MCP Host (the AI application)
- **SIM cards** = MCP Clients (connection handlers)
- **Telephone providers** = MCP Servers (data sources)

Just as you can use any SIM card in a phone with standardized slots, any AI application can connect to any data source through MCP.

Key Components:

- **MCP Host:** The AI application (ChatGPT, Claude, custom apps)
- **MCP Client:** The connection layer within the host
- **MCP Server:** Data source implementations (GitHub, Google Drive, Slack, databases)
- **MCP Marketplace:** Central repository of available servers and integrations

Real-World Application: Expense Calculator Demo

To demonstrate MCP's practical value, consider an expense tracking application built with MCP:

Problem it Solves:

- Maintaining long Excel spreadsheets for budget and expenses
- Converting PDFs/Excel/images to spreadsheets for data evaluation
- Limited to single-person usage in traditional systems

Functionality Enabled by MCP:

- Natural language expense and credit management
- NLP-based editing and deletion of transactions
- Automatic extraction from bank statements (Excel/PDF/images)
- Visual dashboard generation for all expenses and credits
- Multi-user support for tracking household expenses
- Automatic categorization of expenses
- Contextual date understanding ("yesterday," "day before yesterday")

Technical Implementation:

- Remote MCP server (with local server option available)
- SQLite database for data persistence
- FastMCP library for protocol implementation

The New Normal with MCP

With MCP, the Product Manager's workflow transforms completely into a unified AI workflow with:

- Single AI assistant (ChatGPT with MCP integration)
- Connected to multiple tools through standardized MCP servers
- Seamless context sharing across all platforms
- No manual data transfer required
- Real-time access to Slack, Google Drive, databases, GitHub, and more

Benefits:

- **Standardization:** One protocol to rule them all
- **Scalability:** Easy addition of new data sources
- **Security:** Consistent authentication and authorization
- **Maintainability:** Updates to protocol benefit all integrations
- **Developer Experience:** Simple server creation with FastMCP library
- **User Experience:** Truly unified AI assistant

The Path Forward

MCP represents the infrastructure layer for the next generation of AI applications. This protocol enables:

- True AI agent autonomy across platforms
- Elimination of data silos
- Democratization of AI tool development
- Reduced development and maintenance costs
- Enhanced security through standardization
- Seamless user experiences across all tools

The MCP marketplace is growing rapidly, with servers being developed for virtually every major platform and data source. This ecosystem approach ensures that as new platforms emerge, they can be instantly accessible to all MCP-compatible AI applications.

Conclusion

The Model Context Protocol solves one of the most critical challenges in AI adoption: fragmented context across multiple platforms. By providing a standardized way for AI applications to access data sources, MCP enables the vision of a truly unified AI assistant that can understand and work with your entire digital ecosystem.

As AI continues to evolve, MCP will serve as the foundational infrastructure enabling seamless integration, enhanced productivity, and the realization of AI's full potential in transforming how we work.

Key Takeaways:

1. **AI adoption has progressed from experimentation to ecosystem explosion**
2. **Fragmented AI tools created the "problem of silos"**
3. **Context is the foundation of effective AI assistance**
4. **Function calling enabled tools, but created integration complexity**
5. **MCP provides a universal standard eliminating the N×M integration problem**
6. **Real-world applications like the expense calculator demonstrate MCP's practical value**
7. **MCP represents the infrastructure for the next generation of unified AI experiences**

This document summarizes a comprehensive presentation on Model Context Protocol by Ashu Mishra, Technical Product Manager. For more information, visit LinkedIn profile.