Evaluation of Model Context Protocol (MCP) with AI Agents

Date: October 13, 2025
Course: CS-E4660
Research Scope: Evaluating Model Context Protocol (MCP) for AI Agents Interactions

Abstract

The development of artificial intelligence (AI) agents capable of interacting with external tools and services traditionally requires implementing custom integration logic for each API, managing authentication methods, and handling error response formats. The Model Context Protocol (MCP), introduced by Anthropic in November 2024, states that a paradigm shift toward standardized, protocol-driven AI-tool communication. However, empirical research quantifying MCP's transformative impact on development practices and agent capabilities remains absent. This study explorers how the development of AI agents is transformed by MCP adoption. Specifically, the research analyzes whether MCP integration leads to reduced development time and improved AI agent capabilities compared to traditional API integration approaches. The research employs an experimental work with randomized controlled trials comparing MCP-based development (treatment) against traditional API integration (control) using the Arcee Agent model (7B parameter function-calling small language model - SLM). This research provides the systematic empirical evaluation of MCP's transformative impact on AI agent development. Findings will inform software engineering practices, guide protocol adoption decisions, and advance understanding of how standardization shapes development workflows and AI agent capabilities.

Research Problem

How the development of artificial intelligence (AI) agents is transformed by the Model Context Protocol (MCP)?

Research Question

Does Model Context Protocol (MCP) integration lead to reduced development time and improved AI agent capabilities compared to traditional API integration approaches?

The research explores whether MCP adoption (a) transforms development practices by decreasing implementation time, reducing code complexity, and minimizing integration effort, and (b) enhances AI agent operational capabilities measured by task completion rates, tool invocation success, and system efficiency in Arcee Agent model based applications.

Research Problem Context

The landscape of AI agent development is undergoing evolution, with emerging protocols proposing new paradigms for tool integration and context management. Traditional approaches require developers to implement custom integration logic for each external service, manage authentication schemes, and handle error responses. The Model Context Protocol (MCP) presents a approach as follows: A standardized, protocol-driven framework for AI-tool communication. This transformation encompasses multiple dimensions from development workflows and code architecture to AI agent runtime behavior and operational characteristics. However, empirical validation of MCP's transformative states that across these dimensions remains absent, creating uncertainty about the practical effects of protocol adoption.

Research Type

This study adopts a confirmatory research design incorporating exploratory components, aligning with the characteristics of a transformation study. Utilizing a mixed-methods approach, the research integrates quantitative metrics with qualitative analysis to explore the impact of the Model Context Protocol on the development of AI agents. Central to the design is a true experimental framework employing randomized controlled trials (RCTs), which enables assessment of causal relationships. Quantitative data focus on measurable changes in agent performance and operational efficiency, while qualitative inquiry analyzes shifts in development practices and the experiential dimension of transformation among development teams. Together, these methods facilitate a complete evaluation of how MCP reshapes both process and outcome. Apart from the direct enhancements in performance, the study aims to characterize a paradigm shift from customized engineered solutions to a protocol-driven model of integration, which suggests a foundational transformation in AI development methodology.

Confirmatory Approach with Exploratory Elements

The confirmatory approach tests existing theories regarding the benefits of protocol standardization, while the exploratory component evaluates the documented mechanisms through which the Model Context Protocol (MCP) facilitates simplification.

Research Variables

Independent Variable (IV)

Development Approach is operationalized as **MCP Adoption**

Control Condition:

Traditional API Integration is custom, per-service integration workflows

Treatment Condition:

MCP-Based Integration is protocol-driven standardization approach

Dependent Variables (DVs)

Category A: Development Practice Transformation

1. **Development Time Reduction**

   Measured in total hours required for integration

2. **Code Complexity Reduction**

   Quantified via a composite index (scale: 0–100)

3. **Integration Effort Reduction**

   Assessed by the count of discrete integration steps

4. **Developer Learning Curve Acceleration**

   Measured as time (in hours) to reach operational proficiency

Category B: Agent Capability Enhancement

1. **Task Completion Rate Improvement**

   Expressed as a percentage increase in successful task execution

2. **Tool Call Success Rate Improvement**

   Percentage of successful tool/API call executions

3. **Token Consumption Efficiency**

   Average tokens used per task completed

4. **Response Time Improvement**

   Reduction in average time (in seconds) to generate responses

5. **Error Recovery Capability**

   Percentage of successful recoveries from failed or invalid actions

Hypotheses

Note: The symbol → denotes the expected directional impact or expected outcome resulting from MCP adoption.

Development Practice Transformation (H_1–H_4):

1. MCP adoption transforms development workflows → **40–60% time reduction**

2. MCP adoption simplifies code architecture → **45–60% complexity reduction**
3. MCP adoption streamlines integration → **65–75% fewer configuration steps**
4. MCP adoption accelerates skill acquisition → **60–70% faster proficiency**

Agent Capability Enhancement (H_5–H_9):

5. MCP adoption improves tool communication → **15–30% higher success rates**
6. MCP adoption optimizes context usage → **20–40% token reduction**
7. MCP adoption increases task completion → **20–30% higher completion rates**

How Hypotheses Are Calculated?

All effect size predictions follow this methodology:

1. Reasoning from Literature
- Compare MCP to similar standardization interventions in software engineering
- Extract reported effect sizes from peer-reviewed studies
- Apply conservative estimates (lower bound of literature ranges)

2. Task Analysis
- Break down development activities into components
- Estimate savings per component under MCP vs. traditional
- Weight by typical time allocation
- Calculate aggregate effect: Σ(Component_Weight × Elimination_Rate)

3. Pilot Study Calibration
- Measure actual effects in small-scale pilot (n=10-20)
- Compare observed vs. predicted effect sizes
- Adjust predictions if |observed - predicted| > 0.3 standard deviations

Effect Size Formulas

For reductions (H_1, H_2, H_3, H_4, H_6, H_8):

Reduction = [(Control_Mean - Treatment_Mean) / Control_Mean] × 100
Cohen's d = (M_treatment - M_control) / SD_pooled

For improvements (H_5, H_7, H_9):

Improvement = [(Treatment_Mean - Control_Mean) / Control_Mean] × 100
Cohen's d = (M_treatment - M_control) / SD_pooled

Conclusion

This research proposal outlines a confirmatory experimental study to evaluate the integration of Model Context Protocol with the Arcee Agent model. The study tests hypotheses regarding performance, efficiency, and reliability improvements resulting from MCP integration.

References

Arcee Agent https://huggingface.co/arcee-ai/Arcee-Agent

Building AI agents with the Claude Agent SDK
https://www.anthropic.com/engineering/building-agents-with-the-claude-agent-sdk

Developer resources to work with Arcee Agent models on AWS
https://github.com/arcee-ai/aws-samples

Use metrics to understand model performance
https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-metrics.html

AI agents evals for MCP in AIOps
https://www.thoughtworks.com/en-us/insights/blog/generative-ai/AI-evals-for-MCP-in-AIOps

Parnas (1972) https://dl.acm.org/doi/pdf/10.1145/361598.361623

McIlroy (1968) https://www.cs.dartmouth.edu/~doug/components.txt

Basili & Weiss (1984) https://ieeexplore.ieee.org/document/5010301

Prechelt (2000) https://page.mi.fu-berlin.de/prechelt/Biblio/jccpprtTR.pdf

Cognitive Science: Sweller (1988)
https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1202_4
Carroll & Rosson (1987) The paradox of the active user
https://dl.acm.org/doi/10.5555/28446.28451

Tanenbaum & Wetherall (2011) www.distributed-systems.net

Herbsleb & Mockus (2003) https://ieeexplore.ieee.org/document/1205177

Anthropic (2024) https://modelcontextprotocol.io/docs/getting-started/intro

Berkeley Function Calling Leaderboard (2024)
https://gorilla.cs.berkeley.edu/leaderboard.html