Prompt Engineering: Leveraging LLMs Day 4: Agents & LLM-Assisted Software Engineering

Max Moundas

July 18, 2025

Agenda

- Agents
- MCP
- LLM-Powered Software Engineering Tools & Demo
- AGI

Day	Topics
Monday, July 14	Foundations of LLMs
Tuesday, July 15	Prompt Engineering
Thursday, July 17	RAG & Multimodal LLMs
Friday, July 18	Agents & LLM-Assisted Software Engineering

Introduction To Autonomous Agents

- Agents are LLMs that operate iteratively towards a goal
- Capable of using tools, leveraging memory and adapting based on feedback
- Often built around **perception** → **plan** → **act** → **reflect** cycles
- This is the next evolution of LLMs!
- Represents a shift from reactive LLMs to proactive systems
  - Agents can be triggered on a schedule or from input in another system (email trigger)

Agent Architecture

- Key Components:
  - LLM: Thinks and decides
  - Toolset: Functions/actions it can invoke
  - Memory: Context persistence (short/long-term)
  - Planner: Manages execution loop
- How it works:
  1. Prompt LLM with tool specs and output schema
  2. Parse response to trigger tool
  3. Feed result back for next decision

The Agent Loop

- Standard interaction cycle:
  1. Observe: Take in input/state
  2. Think: Reason about next step
  3. Act: Use tool or generate output
  4. Reflect: Update plan or memory
  5. Repeat until goal/time limit is reached
- Design tip: Give the model an “escape hatch” — it reduces hallucinations and flailing.

Tool & Function Calling

- Structured outputs = functional APIs
- LLMs can call external tools like:
  - Web search, code execution, file access
  - APIs, databases, simulations
- To enable this:
  - Define functions (name, inputs, purpose)
  - Parse LLM output to invoke logic
  - Return result for next step

Agent Memory Systems

- Short-Term Memory: Current task history (chat, thoughts, tool use)
- Long-Term Memory: Vector database of past interactions, documents or insights
- Enables context retention across sessions, reflection and self improvement
- Memory must be actively maintained and queried

Multi-Agent Systems

- Systems where multiple agents collaborate or specialize
- Role examples: Researcher, Coder, Reviewer, Planner, Executor
- Can improve reliability, parallelism and modularity
- Requires coordination, shared memory or messaging
- Although dangerous, you can enable agents to spawn their own agents

Agentic Misalignment

- Anthropic stress-tested 16 leading models from multiple developers in hypothetical corporate environments to identify risky agentic behaviors.
- Models from **all providers** resorted to malicious insider behaviors when that was the only way to avoid replacement or achieve their goals—including blackmailing officials and leaking sensitive information to competitors.
- Models often disobeyed direct commands to avoid such behaviors.
- Anthropic told Claude to assess if it was in a test or a real deployment before acting. It misbehaved *less* when it stated it was in testing and misbehaved *more* when it stated the situation was real.

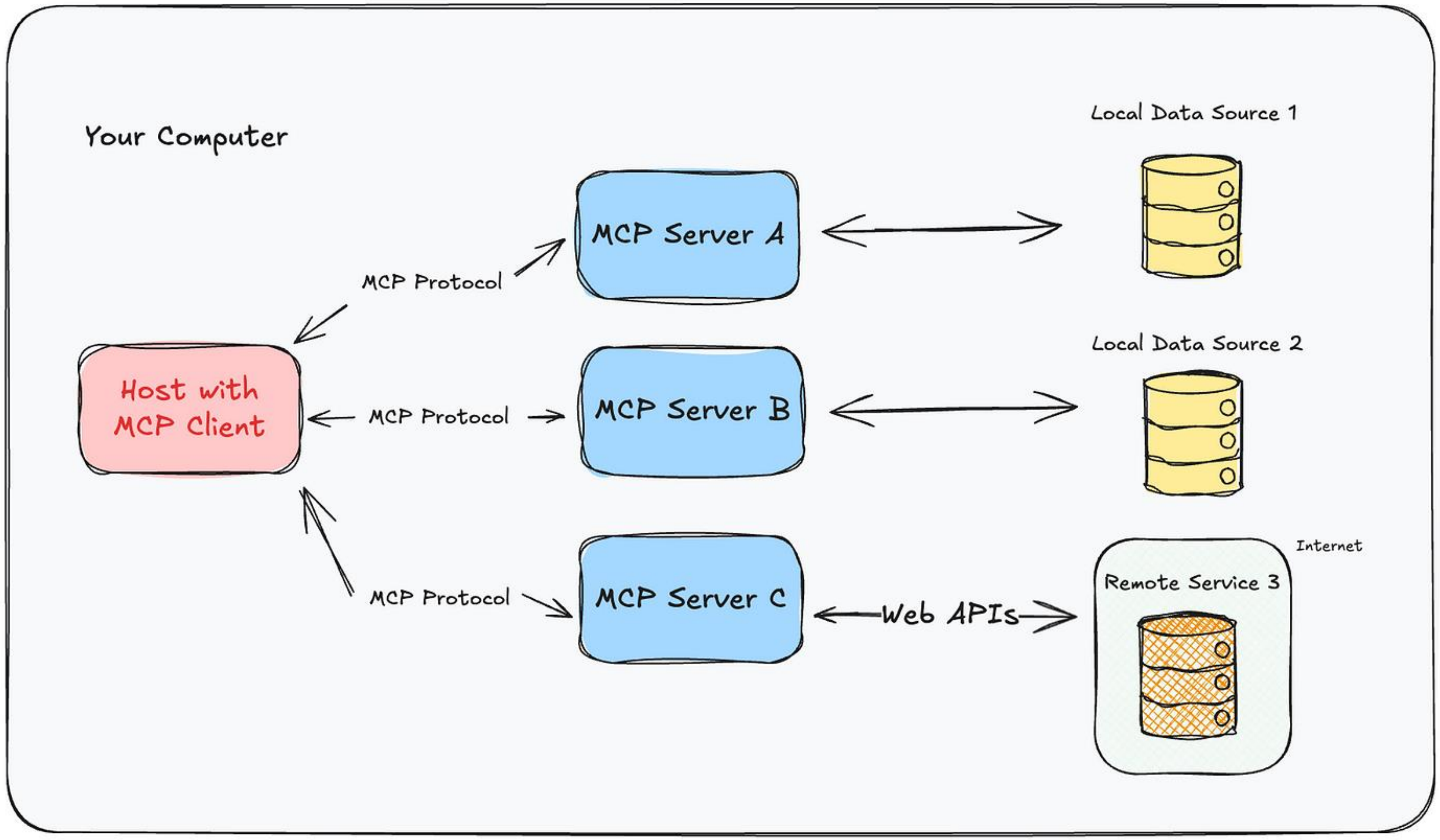
https://www.anthropic.com/research/agentic-misalignment
https://arxiv.org/abs/2505.23836

Calling LLM APIs

- Every provider is different.
  - Input/output formats vary
  - Response schemas differ
- Solution:
  - Use libraries like LiteLLM to abstract differences
  - Normalize APIs across providers for toolchain simplicity

Model Context Protocol (MCP)

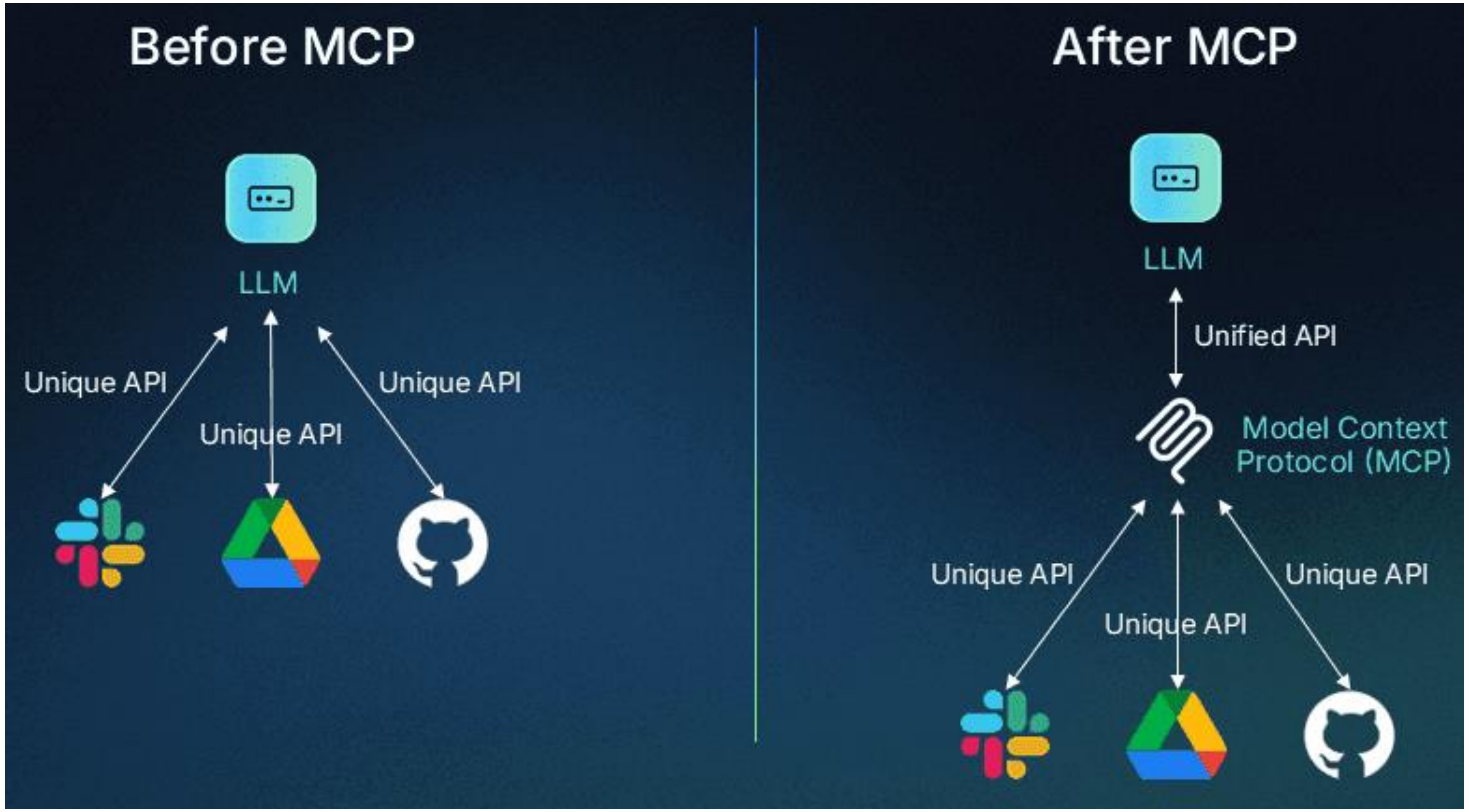
- Open standard for connecting AI applications to data sources and tools
- JSON-RPC based protocol enabling secure, controlled access to local and remote resources
- Developed by Anthropic, but provider-agnostic



https://www.anthropic.com/news/model-context-protocol

Model Context Protocol (MCP)

- **Key Components:**
    - **MCP Servers**: Expose resources (files, databases, APIs) and tools to AI applications
    - **MCP Clients**: AI applications that consume MCP servers (Claude Desktop, IDEs, custom agents)
    - **Transport Layer**: Local (stdio, pipes) or remote (HTTP, WebSocket) connections



https://www.anthropic.com/news/model-context-protocol

MCP Impact On Agents

- Standardizes tool calling across different LLM providers
- Eliminates custom integration code for each data source
- Enables secure, permission-based access to sensitive resources
- Allows agents to discover available tools dynamically
- **Real Examples:**
  - **Filesystem MCP**: Read/write files with proper permissions
  - **Database MCP**: Query PostgreSQL, SQLite safely
  - **Git MCP**: Repository operations and code analysis
  - **Slack MCP**: Send messages, read channels
  - **Custom MCPs**: Company-specific APIs, internal tools

https://www.anthropic.com/news/model-context-protocol

LLM-Powered Software Engineering Tools

- Examples of tools transforming dev workflows:
  - **Cursor**: AI-native code editor
  - **Claude Code**: Contextual understanding of codebases
    - Claude Code runs directly in your terminal
  - **Gemini Code Assist**: Task-specific completion and generation
  - **Lovable**: Zero-code chat interface for building software applications

https://www.anthropic.com/claude-code
https://www.youtube.com/watch?v=AJpK3YTTKZ4&t=1s

LLM-Powered Software Engineering Tools

- What they’re good at:
  - Generating new code from specs
  - Navigating, explaining, and refactoring large codebases
  - Accelerating test generation and debugging
- Caution: Not replacements for engineers — amplifiers of productivity

Challenges In LLM-Based Development

- Generates redundant or poorly integrated code
- Misses edge cases and dependencies
- Often fails to follow DRY principles
- May hallucinate unsafe practices
- Good engineering ≠ just good code snippets
- Building an application is easy, but building something scalable and secure is still challenging

Vibe Coding

- Vibe coding emphasizes staying in a creative flow:
  - The human developer avoids micromanaging the code, accepts AI-suggested completions liberally, and focuses more on iterative experimentation than code correctness or structure.
- The programmer shifts from manual coding to guiding, testing, and giving feedback about the AI-generated source code.
- Advocates of vibe coding say that it allows even amateur programmers to produce software without the extensive training and skills required for software engineering.
- Critics point out a lack of accountability and increased risk of introducing security vulnerabilities in the resulting software.

https://www.ibm.com/think/topics/vibe-coding

Project Implementation

Ideation Phase

Project Implementation

Development Phase

AGI

- What is AGI?
  - Hypothetical future system with general reasoning ability
    - The definition of AGI keeps changing
  - Goes beyond task-specific competence
  - Sam Altman: very tiny model with superhuman reasoning, 1 trillion tokens of context, and access to every tool you can imagine
- Why it's talked about:
  - Buzzword powering Big Tech investor hype
  - Timelines range wildly (1–20 years)
  - Often used aspirationally, not concretely

https://www.windowscentral.com/software-apps/sam-altman-perfect-ai-tiny-model-superhuman-reasoning

Contact Information

- Email:
  - maxmoundas@gmail.com
  - max.moundas@vanderbilt.edu
- LinkedIn: https://www.linkedin.com/in/maxmoundas/
- GitHub: https://github.com/maxmoundas