

Welcome back. You are signed into your member account
bg.....@jaxondigital.com. [Not you?](#)



Me



★ Member-only story

Building Smarter AI Agents: How Sequential Thinking MCP Transforms Complex Problem-Solving

6 min read · Jun 13, 2025



Micheal Lanham

Follow

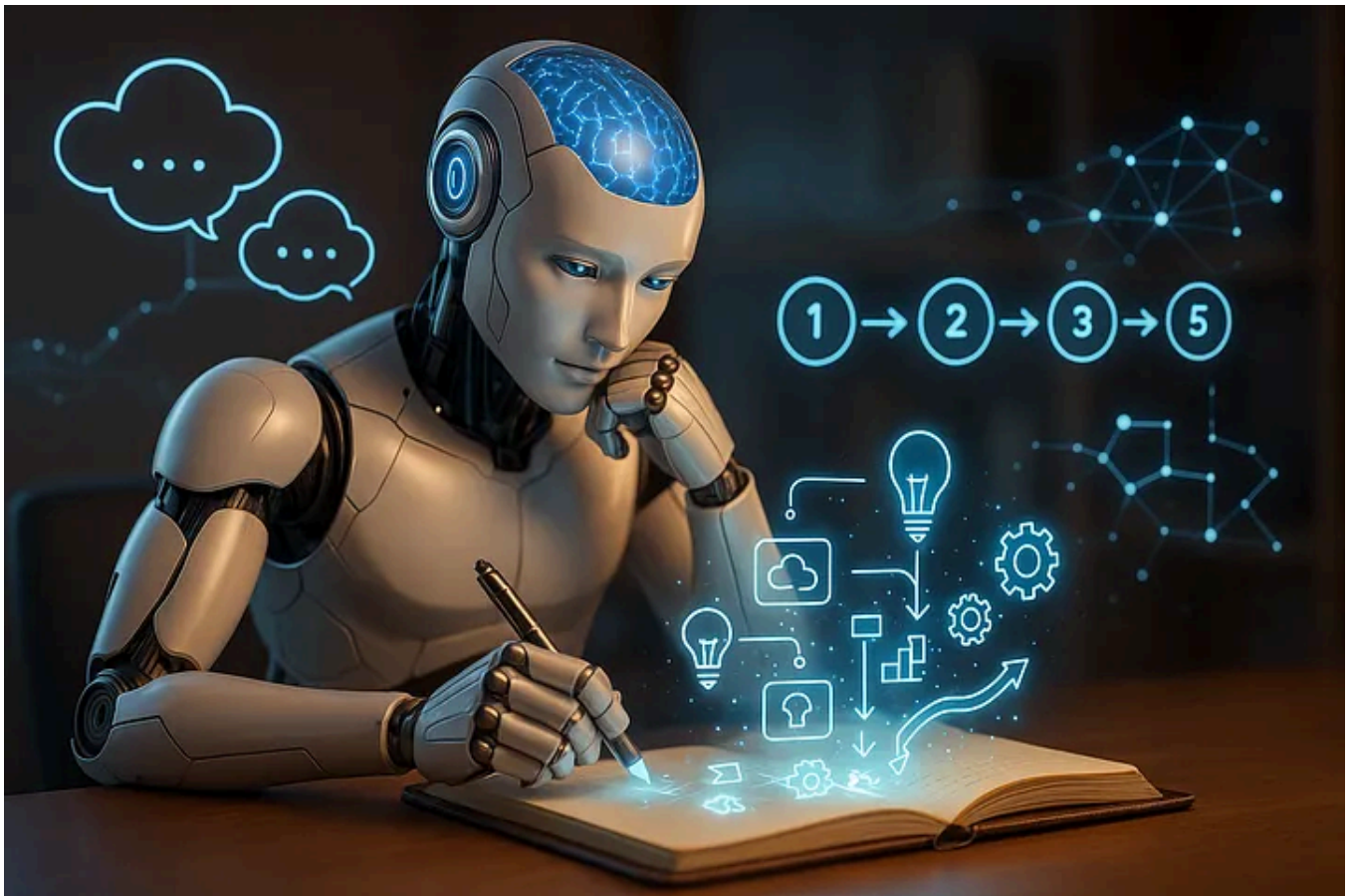


Listen



Share

... More



all images generated with Sora

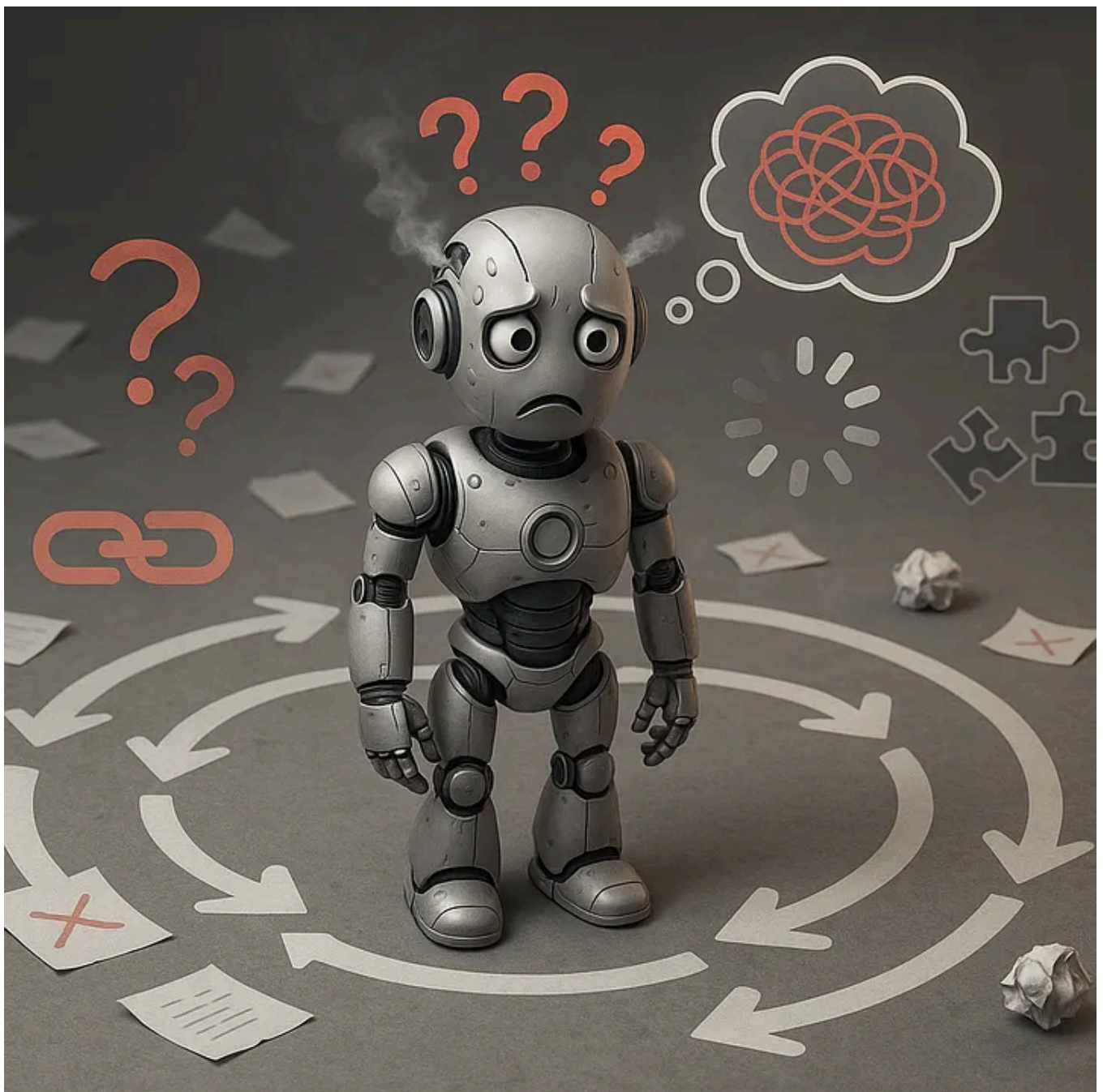
Give your AI agent a structured memory for step-by-step reasoning and watch it tackle complex tasks like a human expert

Have you ever watched an AI agent struggle with a complex problem, jumping between ideas, and not being able to solve it? Today lack the ability to think sequentially until now.

Welcome back. You are signed into your member account
bg****@jaxondigital.com.

The combination of **Sequential Thinking MCP** (Model Context Protocol) with the **OpenAI Agents SDK** is changing how we build intelligent agents. Instead of scattered, inconsistent responses, your agent can now think like a methodical expert: breaking down problems, recording insights, and building on previous thoughts.

Let me show you how this works with a fascinating example: building an AI agent that can plan time travel.



The Problem: AI Agents Think in Circles

Traditional AI agents lack a persistent memory of their own reasoning steps. When faced with a complex problem, they often cycle through the same ideas without learning from previous attempts, leading to repetitive and unreliable outputs. It's like having a brilliant researcher who forgets to take notes.

```
# Traditional agent approach - no memory of reasoning steps
agent_response = "Let me think about time travel...
Actually, let me reconsider...
Wait, what was I analyzing again?"
```

This scattered thinking leads to:

- **Inconsistent reasoning** across complex tasks
- **Lost context** when problems require multiple steps
- **Repetitive analysis** of the same points
- **Unreliable outputs** for mission-critical applications

The Solution: External Memory for AI Thoughts

Here's where Sequential Thinking MCP comes in. Think of it as giving your AI agent a sophisticated

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

- Records each thought with metadata (stage, sequence, importance)
- Organizes **thinking** into structured stages (Problem Definition → Research → Analysis → Synthesis → Conclusion)
- Provides summaries of the entire reasoning process
- Enables **reflection** on previous thoughts

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

Setting Up Your Thinking Agent

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

Getting started is surprisingly straightforward. First, you'll need to install the required packages:

```
pip install openai-agents
pip install mcp-sequential-thinking
```

Then, let's create an agent that can think step-by-step:

```
import asyncio
from agents import Agent, Runner
from agents.mcp import MCPServerStdio

async def create_thinking_agent():
    # Launch the Sequential Thinking MCP server
    seq_server = MCPServerStdio(
        name="SequentialThinkingServer",
        params={
            "command": "mcp-sequential-thinking",
            "args": []
        }
    )
```

```
# C
asy

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

name= ThinkingAgent ,
instructions=(
    "You are a methodical problem-solver. Use Sequential "
    "Thinking tools to organize your analysis step-by-step."
),
mcp_servers=[mcp]
)

return agent, mcp

# Ready to solve complex problems!
```

The Magic: How Thought Feedback Works

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

The real power comes from the **feedback loop** between your agent and its external memory. Here's what happens behind the scenes:

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

Your agent can now:

1. **Record insights** as they develop
2. **Retrieve summaries** to maintain context
3. **Build systematically** on previous thoughts
4. **Avoid repetition** and circular reasoning

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

Real Example: Time Travel Planning Agent

Let's see this in action with a complex scenario: *"Devise a safe plan to travel to the year 2050 and back."*

Here's how our thinking agent tackles this step-by-step:

```
# ... continuing from the previous context
# Define an agent that can use the Sequential Thinking MCP server
agent = Agent(
    name="TimeTravelPlanner",
    instructions=(
        "You are a planning assistant. Think step-by-step, and use the"
        "Thinking tools (thought recording, summary) to organize your p
    ),
    mcp_servers=[mcp] # register the connected MCP server as a tool
)
# Define a user query for the agent to solve
user_query = "Devise a safe and efficient plan to travel to the year 20
result = await Runner.run(starting_agent=agent, input=user_query)
print(result.final_output)
```

The Results: Structured, Reliable Reasoning

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

The difference is remarkable. Instead of scattered thoughts, you get:

Before (traditional agent):

“Time travel is complex... maybe wormholes? Actually, what about... wait, let me think about this differently...”

After (Sequential Thinking MCP):

“Time Travel Plan Analysis:

Problem Definition: Safe bidirectional time travel to 2050 **Research:** Evaluated 3 theoretical methods

Analysis: Relativistic travel most feasible **Synthesis:** Detailed 30-year mission plan

Conclusion: Actionable recommendations with safety protocols”

Advanced Features: Memory Management

With the Sequential Thinking MCP server integrated, the agent can **capture its thoughts and feed them back into its context** in a loop, as illustrated earlier. The MCP server provides three main functions:

- **process_thought** – Records a new thought (with parameters like the thought content, timestamp, and priority). The server may return a status or echo of the recorded thought.
 Welcome back. You are signed into your member account **bg****@jaxondigital.com.**
- **generate_summary** – Produces a summary of the entire thinking process so far on lama.ai (e.g. how many thoughts recorded, and a timeline of stages completed). This is useful for the agent to recall or reflect on the overall progress.
- **clear_history** – Resets/clears the stored thoughts, starting a fresh thinking session on lama.ai.

Why This Matters for Your Projects

This approach transforms AI agents from **reactive responders** into **systematic thinkers**. Perfect for:

- **Strategic planning** agents that need consistent logic
- **Research assistants** handling multi-step analysis
- **Decision support** systems requiring audit trails
- **Educational tutors** that model good reasoning
- **Creative projects** needing structured ideation

Common Pitfalls to Avoid

Don't over-structure simple tasks. Not every query needs five-stage analysis:

```
# ❌ Overkill for simple questions
"What's 2+2?" → [Problem Definition] → [Research] → [Analysis]...

# ✅ Use structured thinking for complex problems
"How should we reorganize our engineering team?" → Sequential analysis
```

Handle connection failures gracefully:

```
try:
    async pass
except ConnectionError:
    # Fallback to regular agent without MCP
    agent = Agent(name="FallbackAgent", instructions="...")
```

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

Getting Started Today

Ready to build smarter agents? Here's your next step:

1. **Install the tools** (`pip install openai-agents mcp-sequential-thinking`)
2. **Start with a complex problem** your current agents struggle with
3. **Implement the thought loop** as shown above
4. **Compare results** with traditional agent approaches

The combination of OpenAI Agents SDK and Sequential Thinking MCP isn't just a technical improvement — it's a fundamental shift toward AI that thinks like we do:

methodically, systematically, and with memory

Your agent

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

ifference.

. . .

Want to dive deeper? The complete code examples and advanced configurations are available in the OpenAI Agents SDK documentation. Have you built thinking agents for your projects? Share your experiences in the comments below.

Tags: #ArtificialIntelligence #OpenAI #AgentDevelopment #MachineLearning #Programming

Micheal Lanham

Ai Agents In Action

Ai Agents

Sequential Thinking Mcp

Mcp Server



Follow

Written by Micheal Lanham

660 followers · 5 following

Micheal Lanham is a proven software and tech innovator with 20 years of experience developing games, graphics and machine learning AI apps.

No responses yet




Bgerby

What are your thoughts?

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

More from Micheal Lanham

 Micheal Lanham


Microsoft Just Solved the AI Agent Problem (And Open-Sourced the Solution)

How the new Agent Framework unifies Semantic Kernel and AutoGen to bring enterprise-grade AI agents from prototype to production

★ Oct 14 🖱 48 💬 3



Welcome back. You are signed into your member account
bg....@jaxondigital.com.


 Micheal Lanham

The 10 AI Agent Platforms Revolutionizing Software in 2025

A hands-on guide to choosing the right framework for building autonomous AI agents—from open-source powerhouses to enterprise solutions

★ Oct 8 🖱️ 78 💬 3



 Micheal Lanham

Building Intelligent RAG Systems with AI Agents: 25 Game-Changing Tips

Your complete guide to creating production-ready Retrieval-Augmented Generation applications that actually work

Welcome back. You are signed into your member account
bg●●●@jaxondigital.com.

○ Micheal Lanham

AI Agents Just Went Enterprise: October 2025 Changed Everything

How Microsoft, OpenAI, Google, and AWS turned “agentic AI” from GitHub experiments into products your company can actually buy

★ Oct 13 🖱 22 💬 2

See all from Micheal Lanham

Recommended from Medium

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

 In Artificial Intelligence in Plain English by Alpha Iterations

Build Agentic RAG using LangGraph

A practical guide to build Agentic RAG with complete project code

★ Oct 19 🤝 184



 Sachh m

Running Complex AI Agents Locally: End-to-End Private Pipeline with MCP and n8n

Disclaimer: 📢 A Note Before You Start Reading

Oct 21 🖱️ 11



Welcome back. You are signed into your member account
bg....@jaxondigital.com.



In Automation Labs by Automation Labs

AI Agent Prompting for n8n: The Best Practices That Actually Work in 2025

Why creating good prompts is not all that complicated



Oct 19 🖱️ 2



In Towards AI by Gao Dalie (高達烈)


RAG is Not Dead! No Chunking, No Vectors, Just Vectorless to Get the Higher Accuracy

Over the past few years, Retrieval-Augmented Generation (RAG) has become a standard feature in nearly all AI applications. However, RAG has often been criticized for its complexity and the need for chunking and vectorization. In this article, we explore a new approach to RAG that eliminates these requirements, resulting in higher accuracy and simpler implementation.

Welcome back. You are signed into your member account
bg....@jaxondigital.com.

★ Oct 17 🤝 603 💬 6



 KailashPathak


How Google Chrome DevTools MCP Empowers AI Debugging with Real Time Performance Insights

Google has released Chrome DevTools MCP, a Model Context Protocol (MCP) server that allows AI coding assistants to directly interact with a...

Oct 8 🤝 19 💬 2




Welcome back. You are signed into your member account
bg....@jaxondigital.com.

 NocoBase

Top 11 Open Source No-Code AI Tools with the Most GitHub Stars

We've compiled a list of 11 open-source no-code tools powered by AI to help you quickly find the platform that best suits your needs.

Oct 21  75



See more recommendations