AI Agent Architectures: The Ultimate Guide With n8n Examples

5 Single AI Agent Architectures
3 Multiple AI Agent Architectures
3 Best Practices while building AI Agents

Table of Contents

2025 is the year of AI agents.

I see many abstract agent architectures on social media. But no one explains how to build them in practice.

So, here's a complete guide with examples prepared in n8n. It will help you learn by doing.

We discuss:

1. Five Single AI Agent Architectures
2. Three Multiple AI Agents Architectures
3. Three Best Practices When Building AI Agents

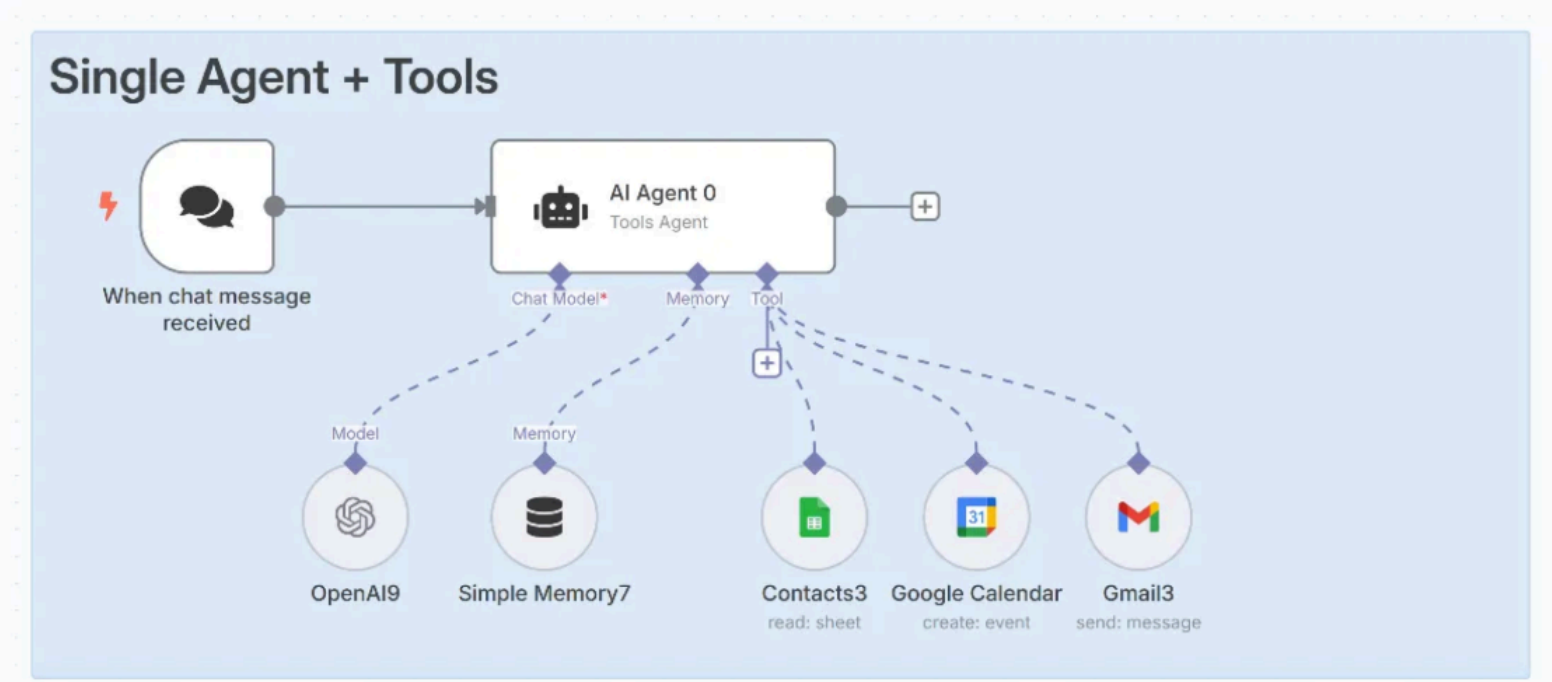
We don’t discuss:

- Fancy terms to memorize
- Things irrelevant when working on AI products

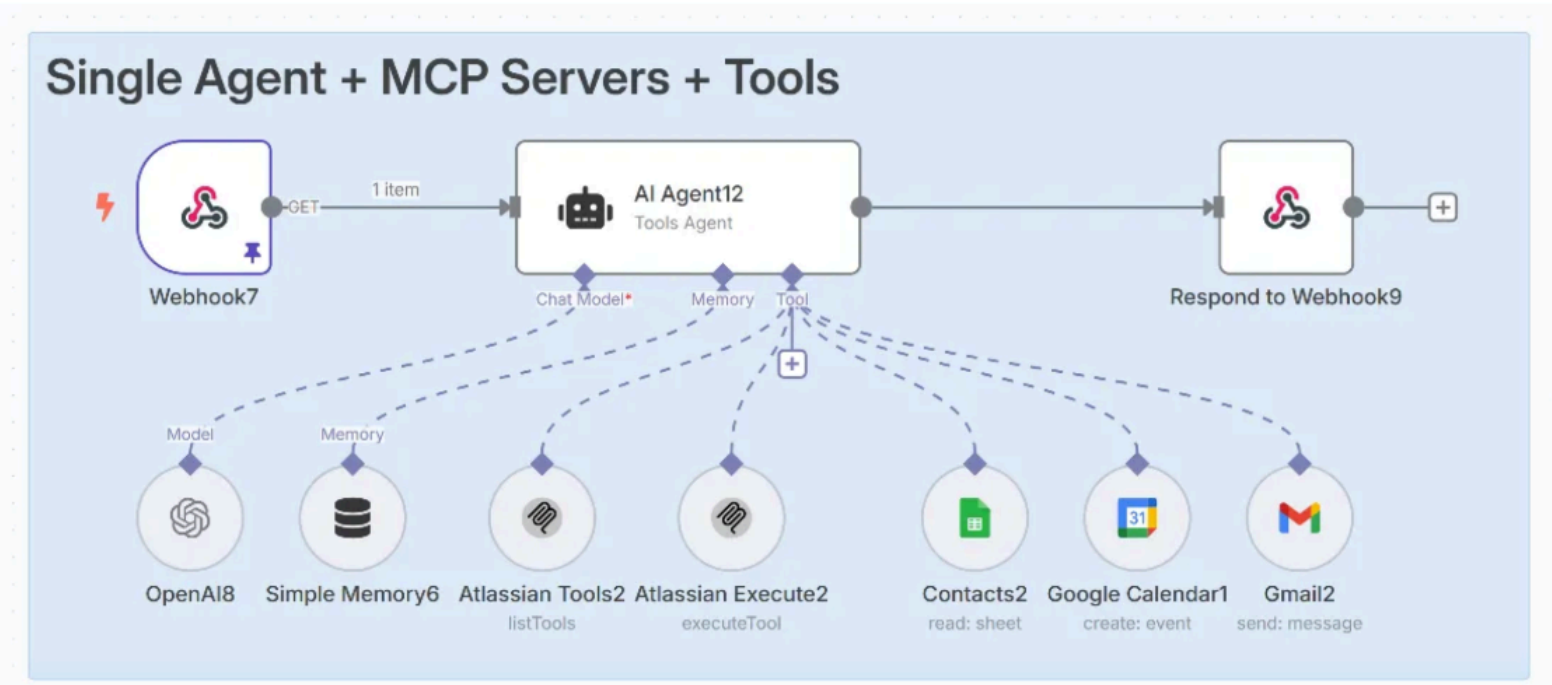
1. Five Single AI Agent Architectures

A picture is worth a thousand words. I prepared 5 configurations you can analyze:

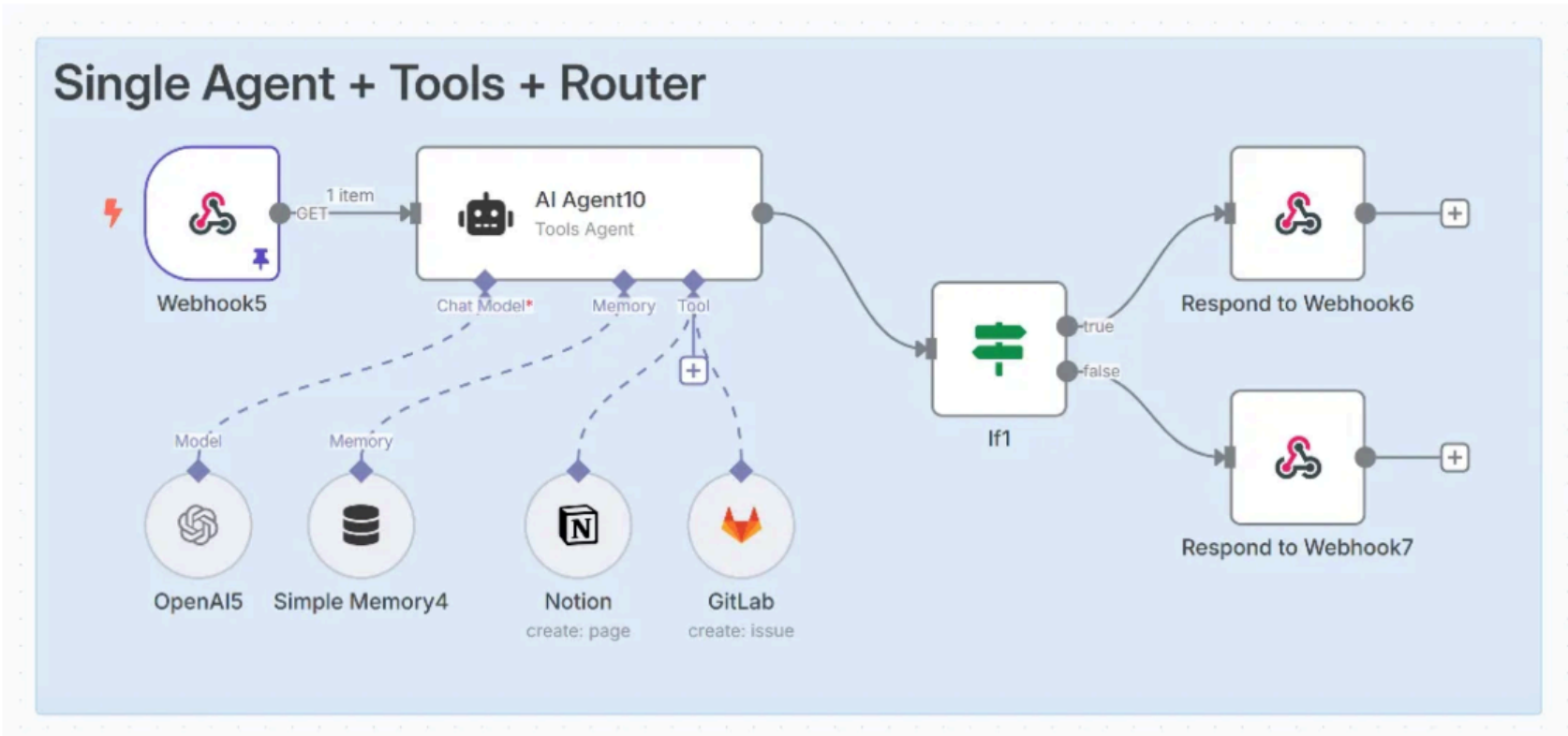
1. AI Agent using tools (based on a chat message, it can plan its actions: access my contacts, send emails, or send invitations)



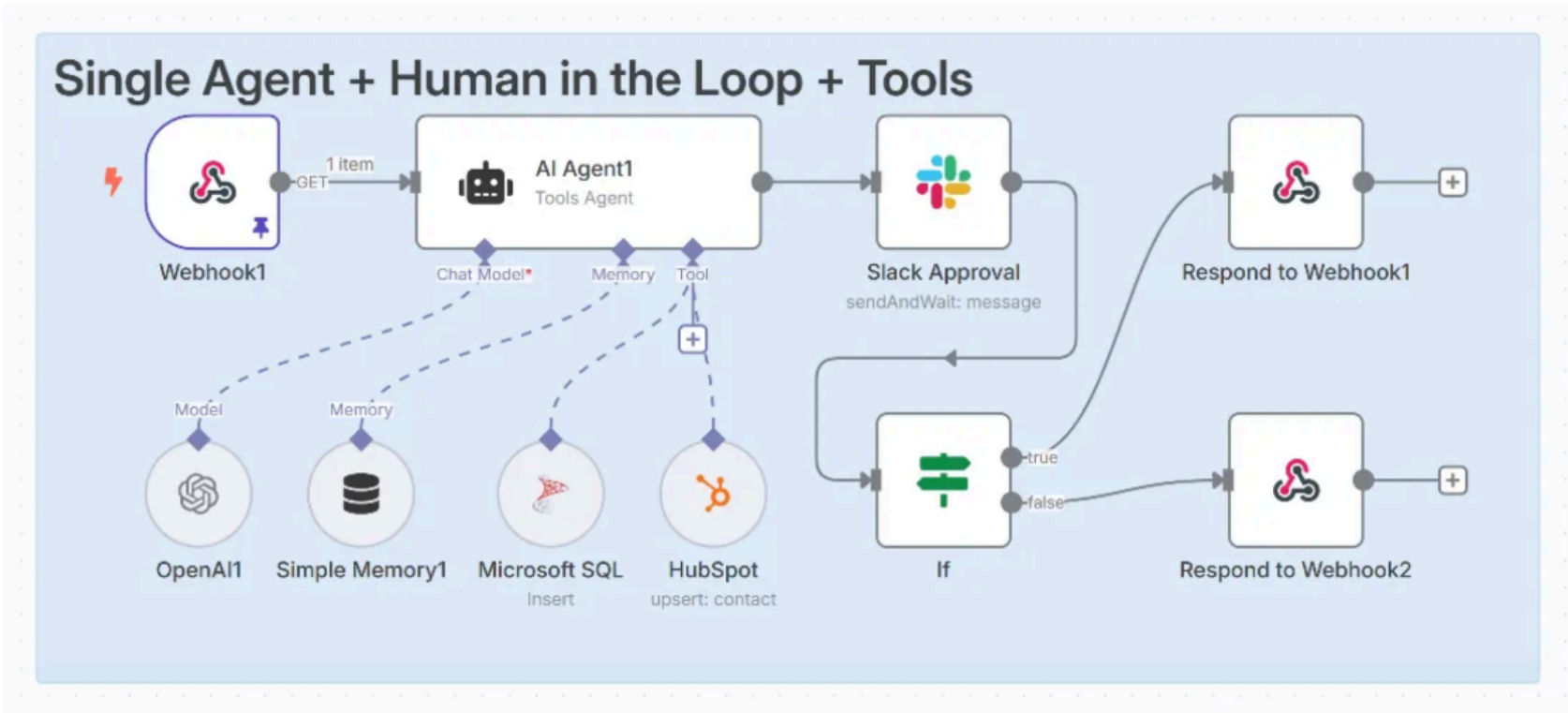
2. Mixing tools with MCP servers (initialized by another app through a webhook with an MCP server for Atlassian and ready-to-use tools for other interactions)



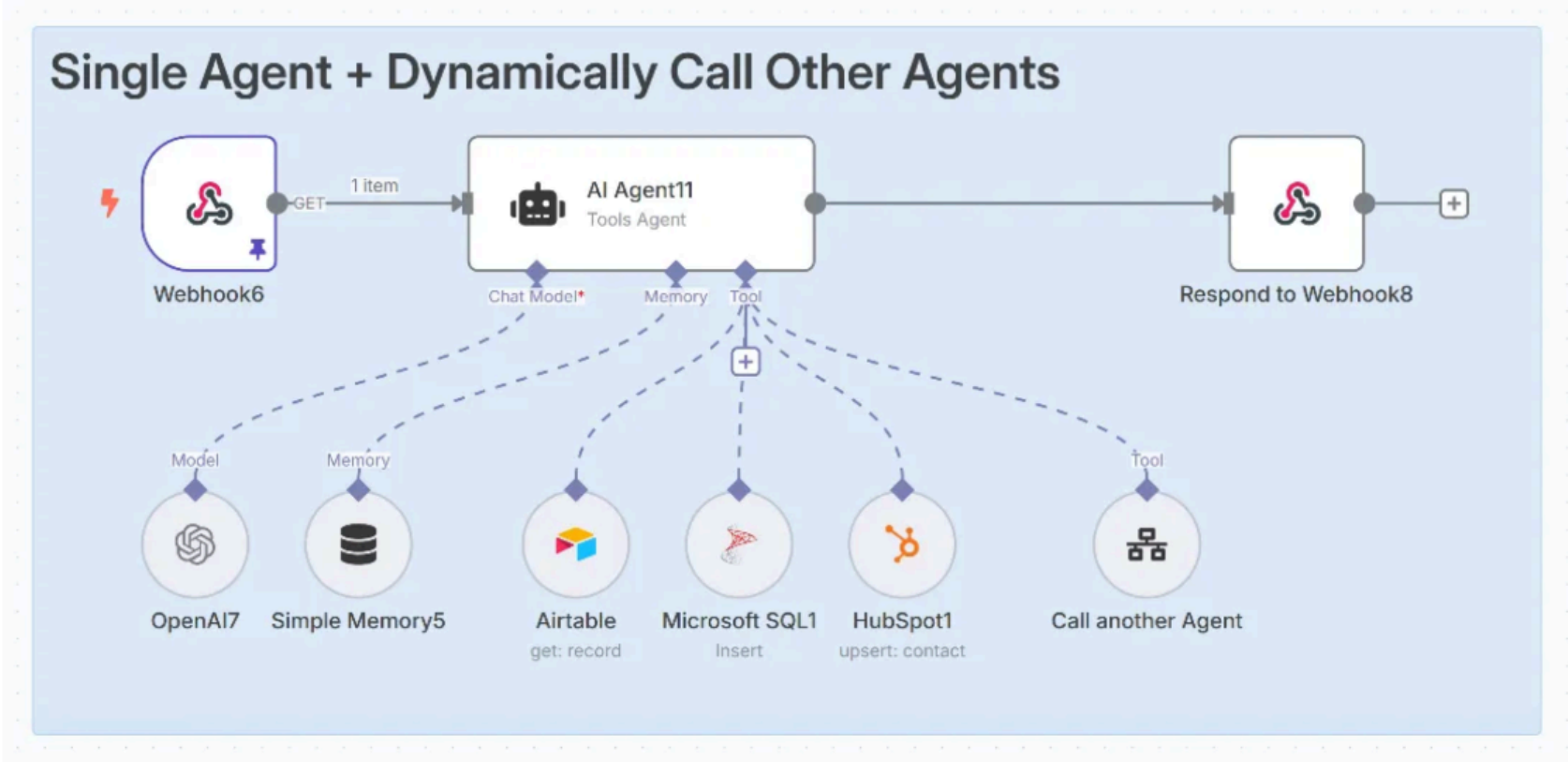
3. Agentic workflow with a router (a fancy name for a condition)



4. AI Agent with a human in the loop (asking for a Slack approval)



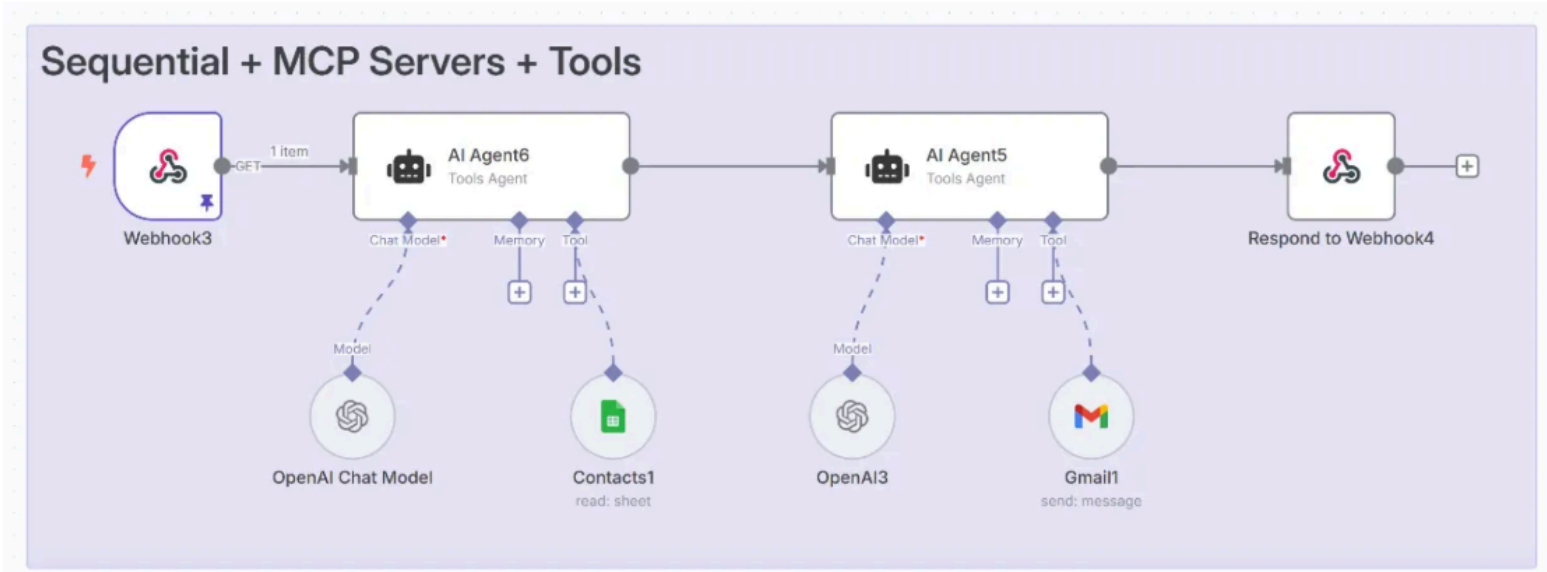
5. Dynamically calling other agents (an agent autonomously decides whether it needs to call another AI)



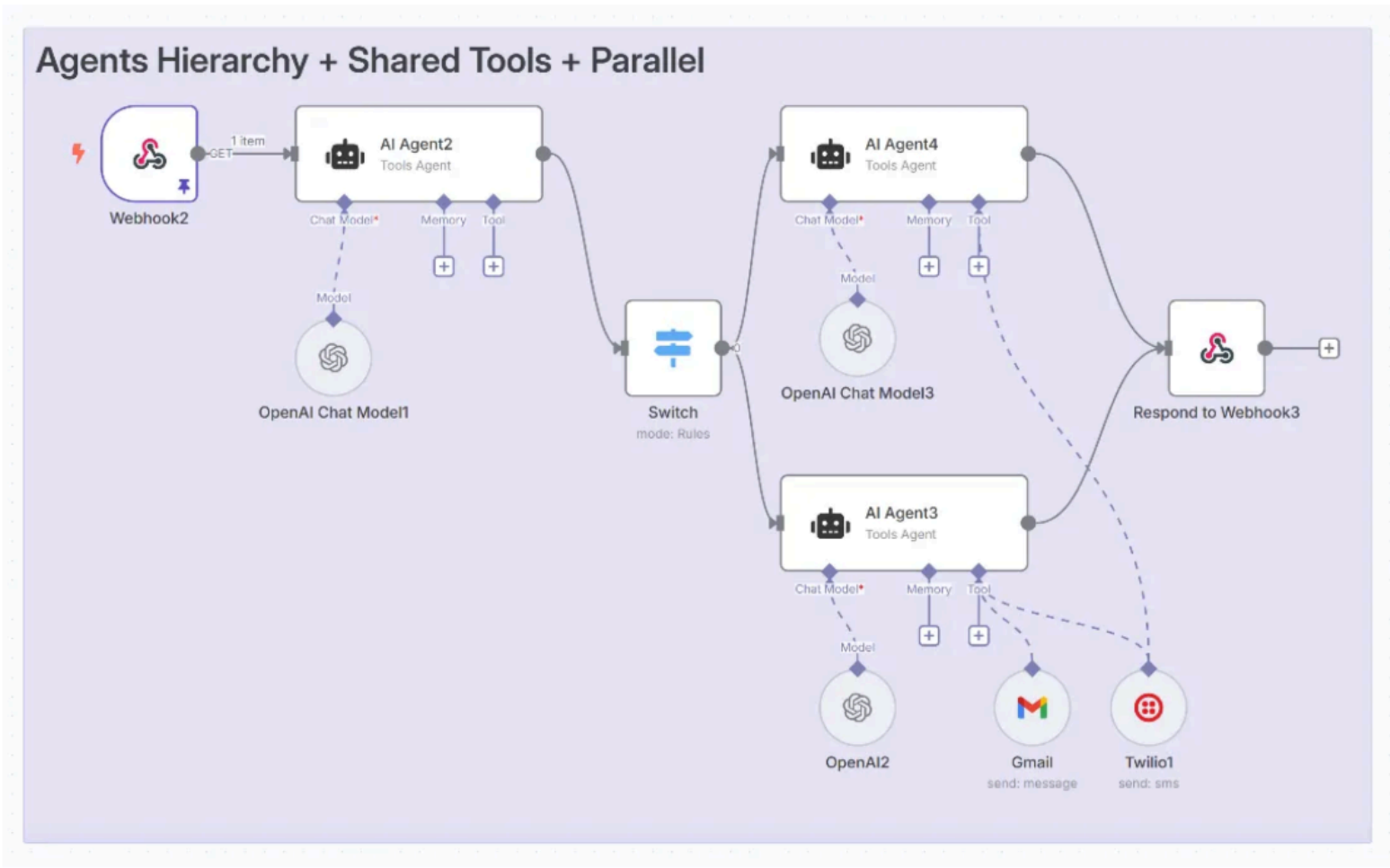
2. Three Multiple AI Agents Architectures

Selected configurations you can analyze:

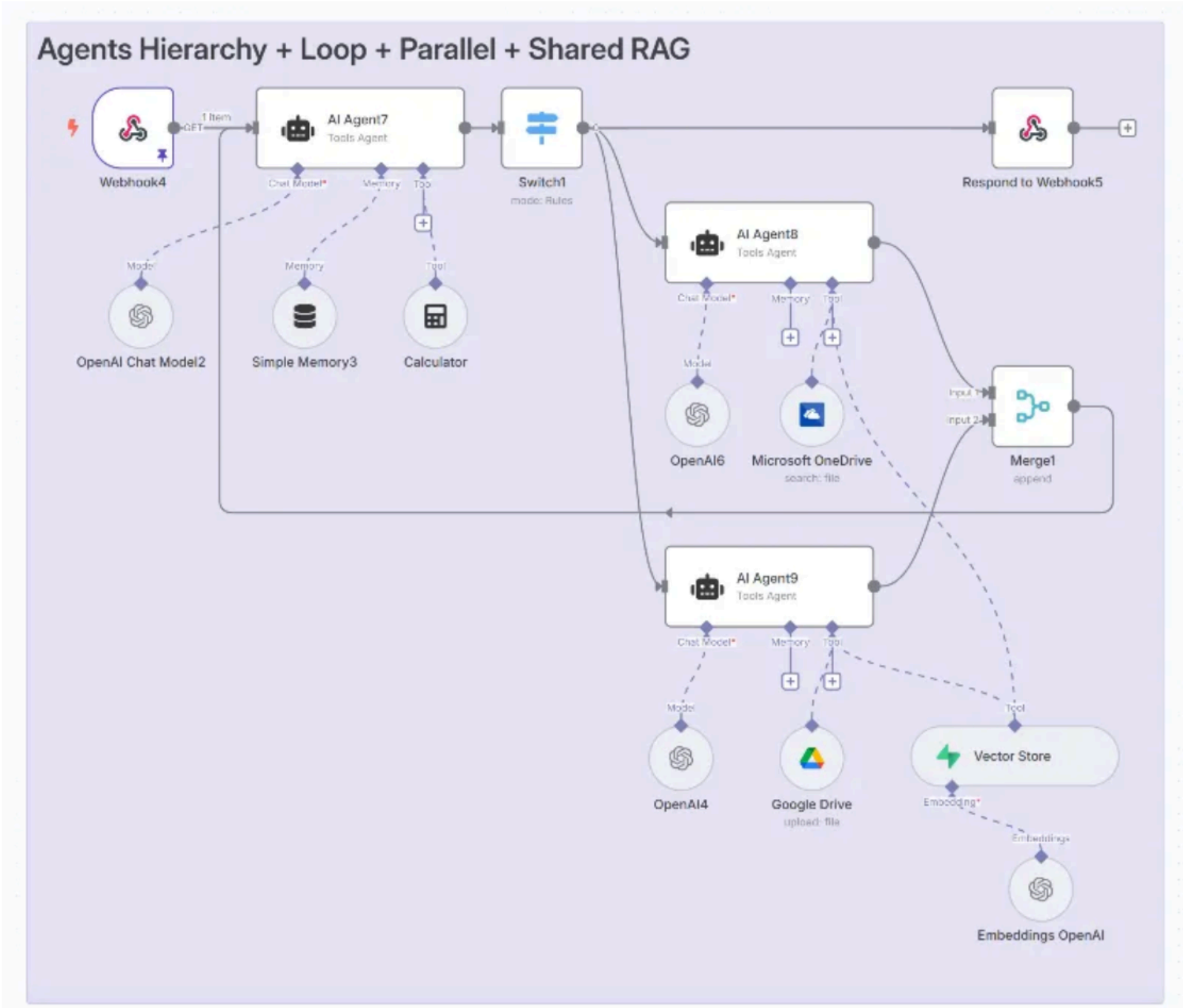
1. AI Agents working sequentially (the first agent reads my contacts to clarify the prompt, another one sends an email to the identified person)



2. Agents' hierarchy with parallel execution and shared tools (Twilio)



3. Agents’ hierarchy with a loop and shared RAG (they perform search in parallel, next the data is merged)



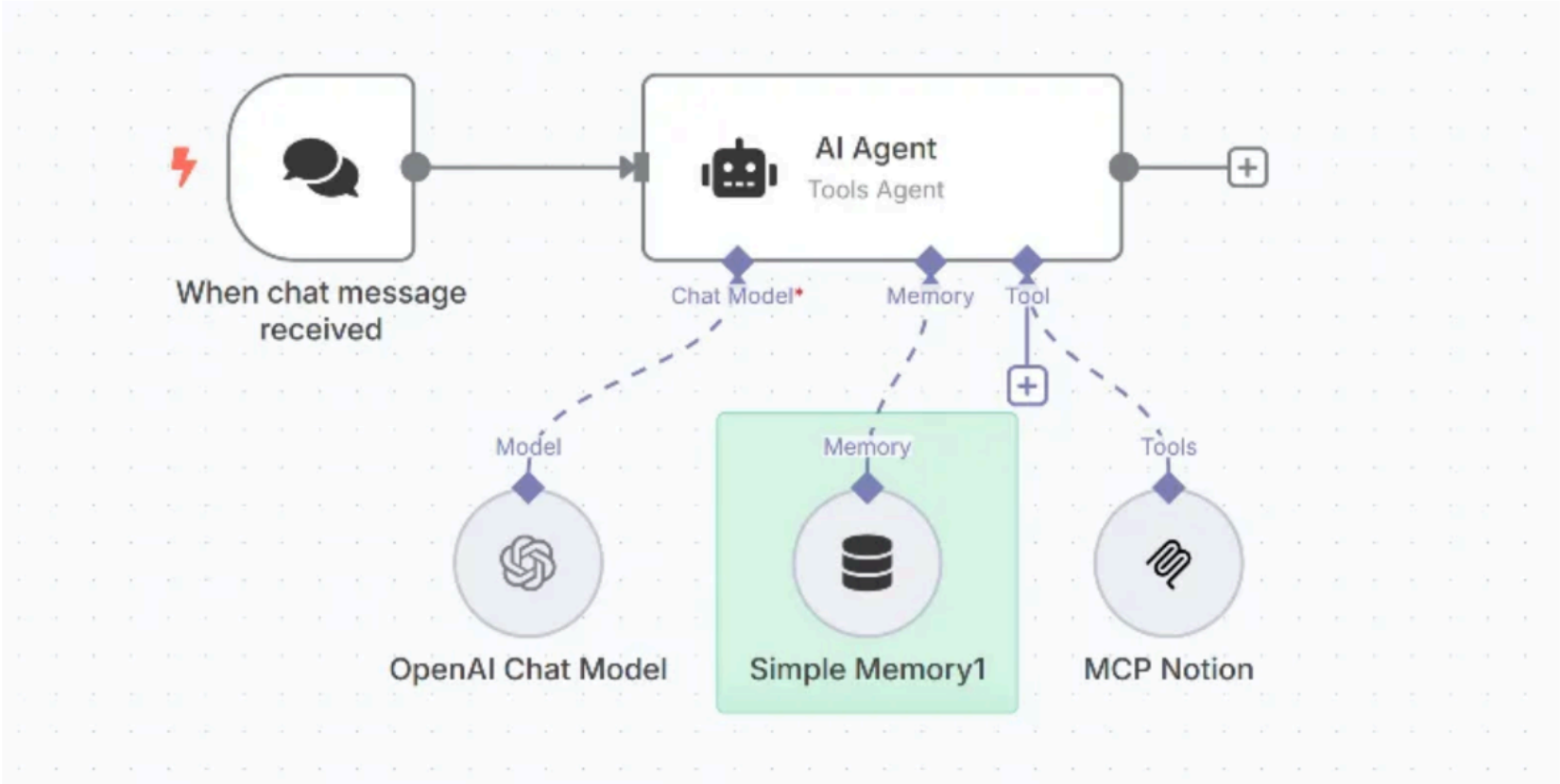
3. Three Best Practices When Building AI Agents

Here's what works best for me:

Best practice 1: Add memory so the agent can track its progress

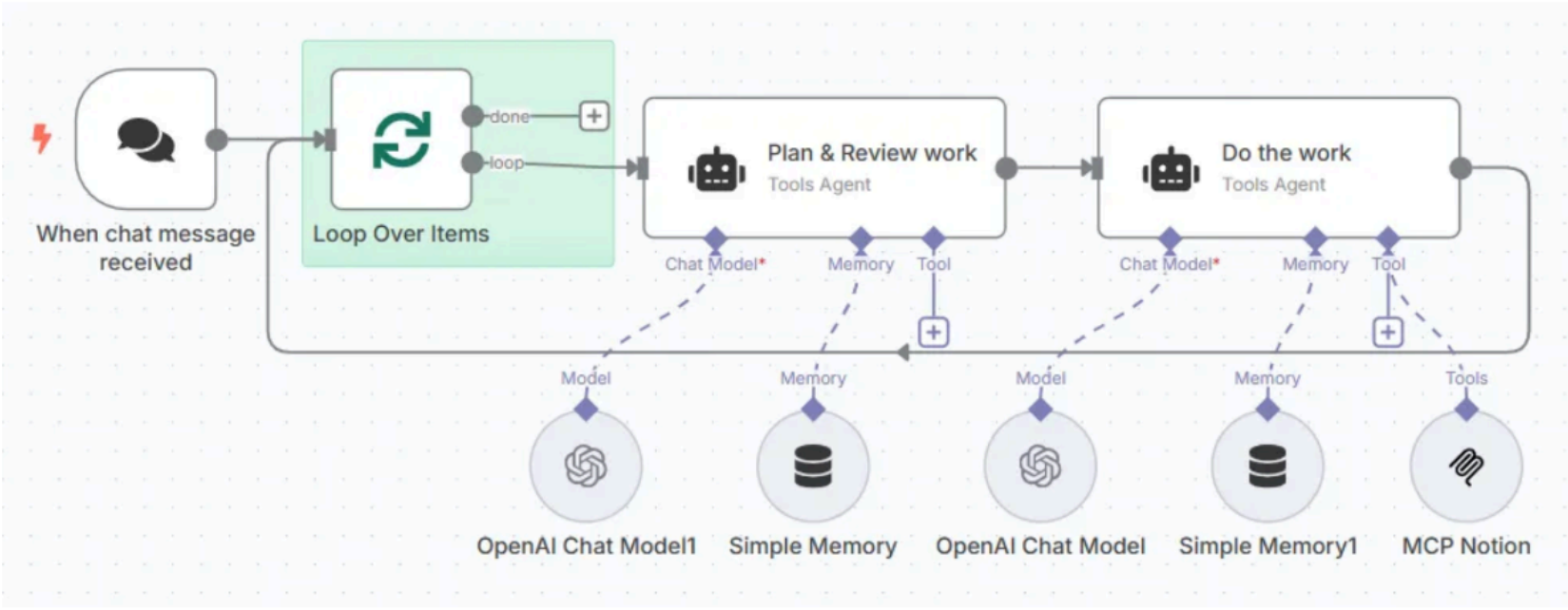
When building agentic workflows, the first thing we often want to do is to ask the agent to **prepare a plan**. For non-reasoning models to follow it, you need to add a memory.

I encourage you to review the "Simple Memory" node configuration in n8n. It can be shared between user requests and even between agents.



Best practice 2: Use a loop to better control complex processes

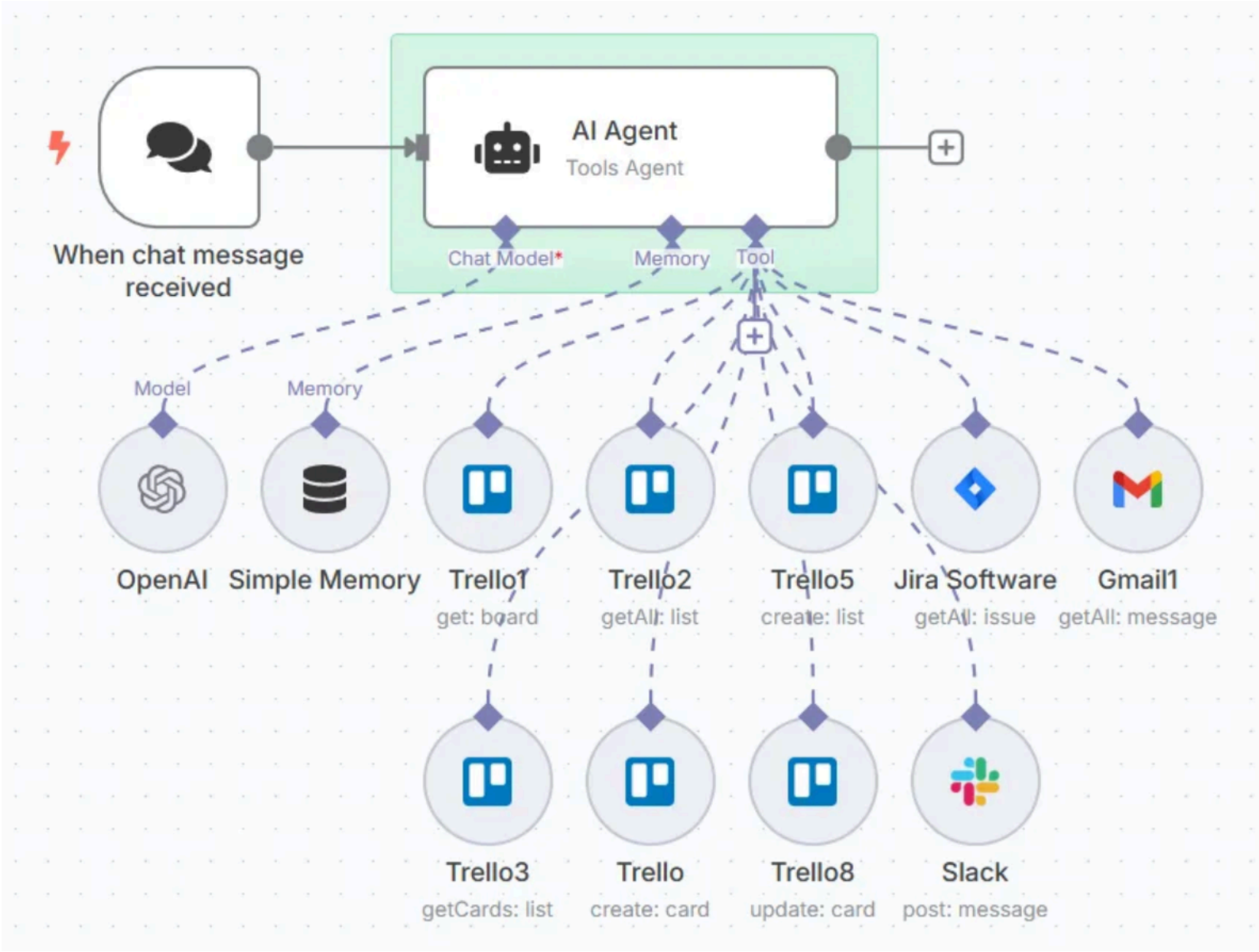
I have found that for complex agentic workflows, creating a loop with a direct split of responsibilities works much more reliably than asking a single agent to plan, perform the work, and inspect its progress.



Best practice 3: Suggest common tool usage patterns

Some MCP servers offer dozens of tools that need to be executed in a specific order. I have found that in those cases default tool descriptions are often not enough.

An example agent with multiple tools:



Interested in more content like this?

Follow me :

OM NALINDE

linkedin.com/in/that-aum