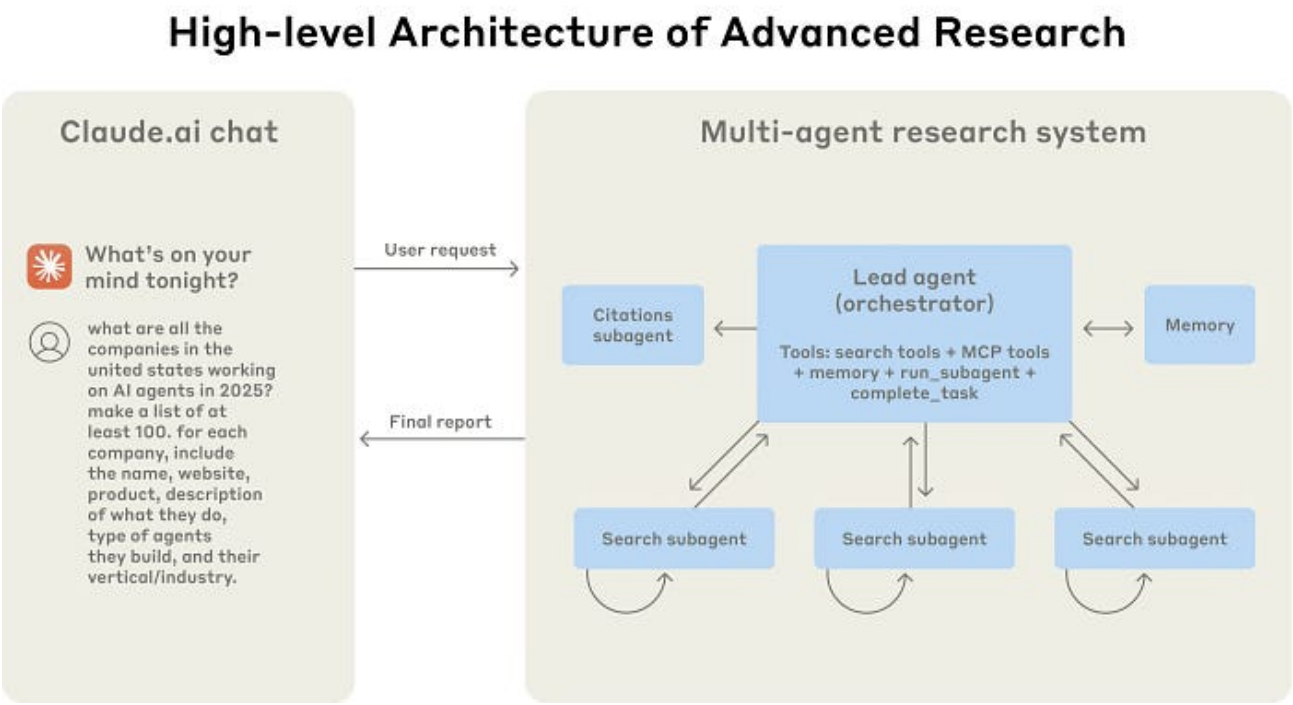
A Prompting Playbook for Building Multi-Agent AI Systems

1. Think Like an Agent

Yashwanth Sai • 7 min read • 2025-08-20

https://medium.com/@theyashwanthsai/a-prompting-playbook-for-building-multi-agent-ai-systems-2f33bd3c2bbb



Lessons from Anthropic's Latest Study

If you cant read this, Click here

Note: This is a small part from my upcoming book on AI Agents. These insights are well expanded in the chapter 3 of the book. I'm sharing a lot in public as we finish the book! Stay tuned because we have a lot of cool stuff to share!

In June 2025, Anthropic revealed the inner workings of their multi-agent research system powered by Claude models. The system was designed to handle complex, open-ended queries that require dynamic planning, web searches, synthesis, and coordination.

Taken from Anthropic's blog

Unlike traditional Retrieval Augmented Generation (RAG), which pulls a few static documents, this system dynamically spawns multiple Claude agents, each tasked with different aspects of a research problem. It's an orchestrator-worker model, where one lead agent thinks through the entire plan and then creates parallel subagents that each explore their own thread.

This system delivered a 90% improvement in research accuracy over a single-agent setup. But the magic wasn't just in the architecture. A large part of the system's performance came from how they prompted the agents.

In this post, I'll walk you through every prompting strategy Anthropic used to make this system work. Whether you're building your own agent workflows or just exploring the frontiers of AI, these lessons are essential.

This is where it all begins.

At Anthropic, prompt engineering wasn't just about writing better sentences. It started with **developing empathy for the agent**.

The team ran agent tasks through the Claude Console and **watched every step**. They looked at how agents made decisions: which tools they used, how they phrased searches, when they stopped reasoning, and where things went wrong.

What they found was often surprising. Agents would:

• Perform redundant searches, even after finding a good answer

• Misunderstand task boundaries

• Use general tools when a specific one was available

• Keep reasoning long after they had a complete solution

So they shifted the mindset: *"If I were this agent, what would I do at each step? What might confuse me?"*

This insight watching, reflecting, iterating became the foundation for every prompt they wrote.

Takeaway: Don't write prompts in isolation. Watch your agents work, understand their blind spots, and iterate like you're debugging code.

2. Delegate with Surgical Precision

Anthropic's lead agent wasn't just a "manager." It was the system's brain, responsible for breaking down a user query and distributing subtasks to subagents.

Early on, prompts looked something like this:

"Investigate the global semiconductor shortage."

What happened?
Three agents did the same search. One gave political context. One talked about chip design. One rambled about supply chains.

No coordination. No alignment. No clue.

To fix this, they rewrote the prompts to include:

• A **clearly scoped subtask** (not just "research," but *"Find recent U.S. legislation related to semiconductor manufacturing"*)

• **Tool instructions** ("Use web search, limit to results from past 12 months")

• **Explicit output format** ("Return a bulleted list with links")

• **Negative instructions** ("Do not include global policies or history before 2022")

Takeaway: When writing multi-agent prompts, define the task *and* the edges of the task. Clarity in what not to do is just as important as the goal.

3. Scale Effort Based on Complexity

One thing Anthropic noticed early: their agents didn't know when to stop. A simple fact-check would trigger 10 agents and 30 web calls. A research task that needed depth was barely touched.

The fix? **Prompt with heuristics** for effort.

They added reasoning like this to the lead agent's instructions:

• If the query is factual, use 1 agent and no more than 3 tools

• If the task involves comparisons, use 2-4 agents with scoped tools

• If it's an open-ended research task, spawn 10+ agents and assign roles carefully

This change reduced both compute and confusion.

Takeaway: Your agents shouldn't brute-force everything. Prompt them to match effort with complexity. It saves time, compute, and your sanity.

4. Teach Agents How to Choose Tools

Another common issue: agents would default to familiar tools, even when better ones were available.

Claude models were sometimes using web search for internal company docs, or querying APIs that returned nothing, all because they weren't prompted with **tool selection heuristics**.

To fix this, Anthropic added rules like:

• *Always scan available tools before choosing*

• *Prefer narrow, specific tools over general-purpose ones*

• *Use internal tools for company data; external tools for research*

• *Don't use a tool if your context shows it's out-of-date or irrelevant*

They even created a **tool-tester agent** that ran tools in sandbox mode and rewrote their descriptions if agents were misusing them.

Takeaway: Your prompt shouldn't just list tools - it should teach the agent how to choose the right one, like a smart operator with a toolbox.

5. Let Agents Improve Their Own Prompts

This was one of the most fascinating ideas Anthropic explored: **self-debugging agents**.

If a task failed or got a poor result, they didn't manually fix the prompt. They spun up a **tool-tester agent** whose job was to:

• Re-run the failed task

• Identify what part of the prompt or tool call failed

• Rewrite the prompt, function schema, or tool description accordingly

The result? **40% reduction in task completion time**, simply by letting the model debug its own prompt interface.

Takeaway: Agents can prompt themselves better than we think. Build feedback loops into your workflows to let them course-correct.

6. Start Broad Before You Go Deep

One of the biggest issues Anthropic noticed was agents becoming too specific, too fast. Early prompts led to narrow searches like *"latest US chip export ban"* without understanding the broader context. This made agents brittle they'd miss relevant angles or fixate on poor results.

To fix it, they rewrote prompts to encourage **broad-first thinking**:

• Start with general queries.

• Get a sense of the landscape.

• Only then drill down into specifics.

This helped agents explore alternatives, discover gaps, and build better context for downstream steps.

Takeaway: Add a line like *"Begin with a broad overview before narrowing down."* It makes agents more flexible and less error-prone during research.

7. Scaffold Thinking with Reasoning Prompts

Agents often treat each tool call as a one-shot answer. That's dangerous when the task requires reflection or iteration.

Anthropic used lightweight "thinking prompts" to build reasoning scaffolds. Before acting, agents were told to plan:

• What am I trying to find?

• What tools do I need?

• What's the expected outcome?

After acting, they'd reflect:

• Did this result help?

• What's the next best step?

This created a loop of **Plan !' Act !' Reflect. I**mproving both accuracy and adaptability.

Takeaway: Add prompts like *"Think through your approach before acting"* and *"Reflect on whether your step achieved the goal."* These make agents feel more intentional and less robotic.

8. Prompt for Parallel Execution

Early versions of Anthropic's system ran agents and tools sequentially. Slow, inefficient, and redundant. A task that could've been done in 2 minutes took 20.

They rewrote the lead agent's prompt to encourage **parallelism**:

• Spawn multiple subagents at once.

• Let each subagent make multiple tool calls in parallel.

This simple change cut task time dramatically and improved coverage.

Takeaway: In multi-agent setups, encourage concurrency. Prompt your lead agent to break tasks into parallel subtasks instead of doing them one-by-one.

9. Heuristics Work Better Than Hard Rules

Anthropic learned that rigid, step-by-step instructions made agents brittle. Instead, they gave **flexible heuristics**:

• “Compare sources before trusting.”

• “Stop if a high-confidence answer is found.”

• “Prefer recent, high-quality documents.”

This gave agents room to reason and adapt when things didn’t go as planned.

Takeaway: Don’t overconstrain your agent. Use heuristics to guide behavior while letting the model make smart decisions in dynamic situations.

10. Use LLMs to Evaluate Each Other

How do you know if an agent’s answer is good? Anthropic prompted **separate Claude agents as judges**. They evaluated responses across:

• Factual accuracy

• Source quality

• Completeness

• Tool usage

They found that a single LLM call with a simple evaluation prompt worked better than complex rubrics.

Takeaway: Use one agent to grade another. A simple evaluation prompt can drastically improve quality control in multi-agent workflows.

11. Prompt for Collaboration, Not Just Completion

In early versions, agents worked in silos. No communication, no alignment. It broke the system’s coherence.

Anthropic fixed this by prompting:

• Defined roles and ownership

• Clear interfaces for passing results

• Coordination rules (e.g., “confirm with Agent B before concluding”)

This transformed a group of solo agents into a **team**.

Takeaway: Design your prompts with collaboration in mind. Define how agents interact, share context, and depend on each other.

Final Thoughts

Prompting in multi-agent systems isn’t about getting a model to say the right thing. It’s about shaping behavior across multiple autonomous agents that operate, plan, adapt, and collaborate.

Anthropic’s experience shows how powerful prompting can be when treated as a systems design challenge. From delegation to tool usage, from search strategy to evaluation, every part of their research system was scaffolded by thoughtful, iterative prompt engineering.

Whether you’re building agents, exploring research automation, or just learning how to work with LLMs more effectively, these lessons are a blueprint for success.

Other mentions by Author

• medium.com | Written by Yashwanth Sai