Creating a Multi-Agent Flow

Building specialized AI agents that work together

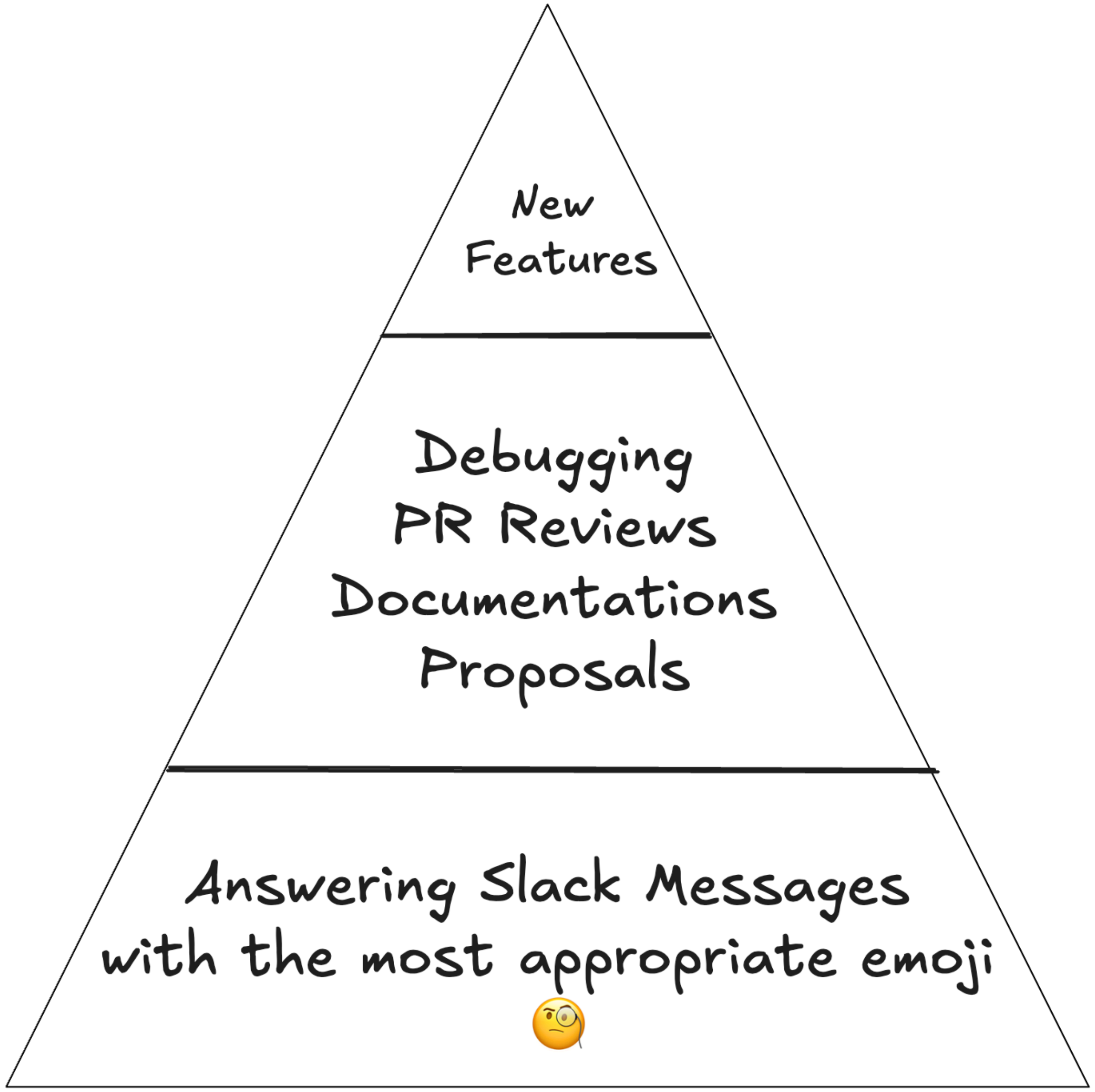
Jhonatan Salazar

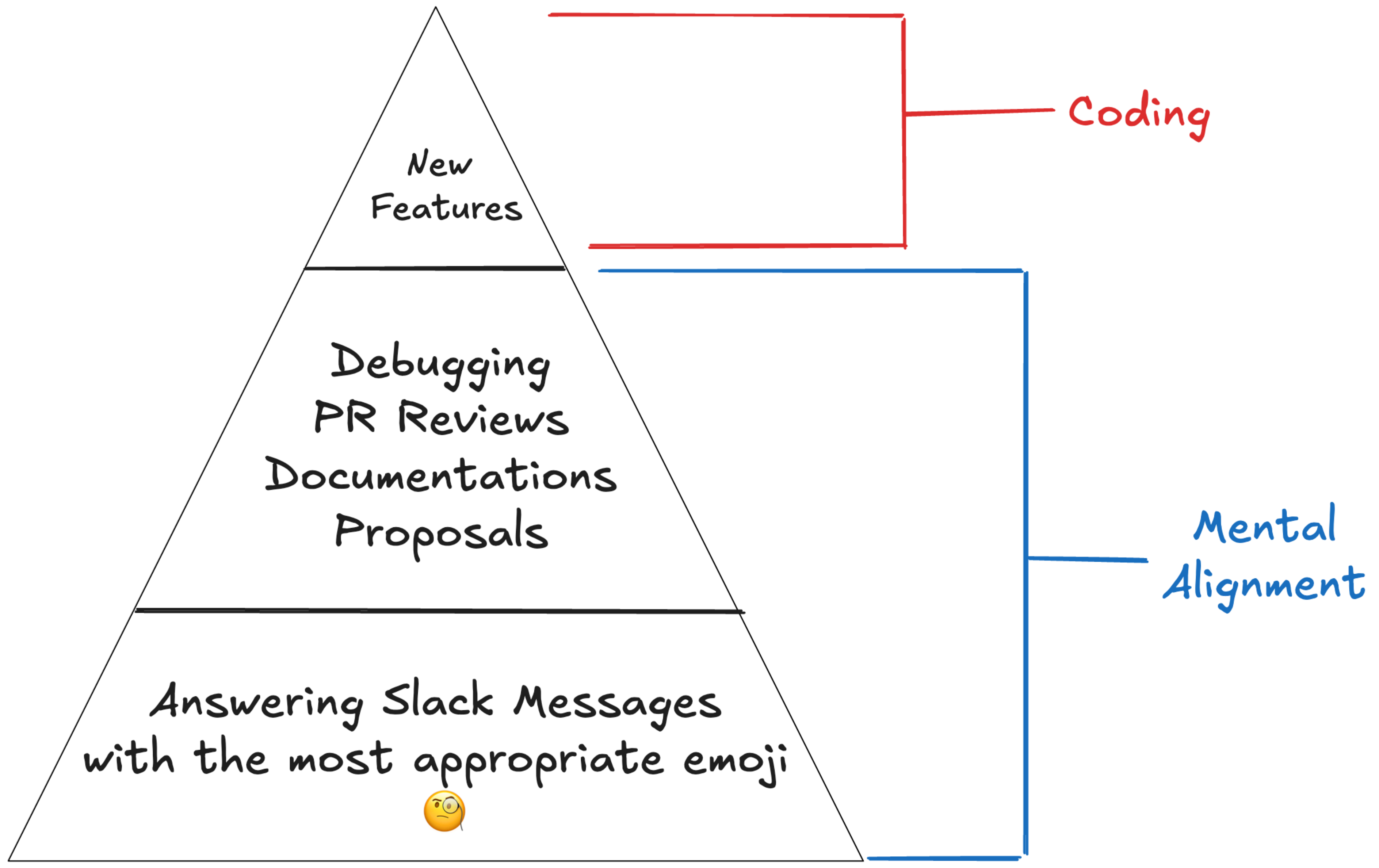
SacTech '25

What We'll Cover

1. What "Agents" actually are
2. How we use agents today & their limitations
3. Building your own custom agents
4. Specialized agents working independently
5. The power of multi-agent flows

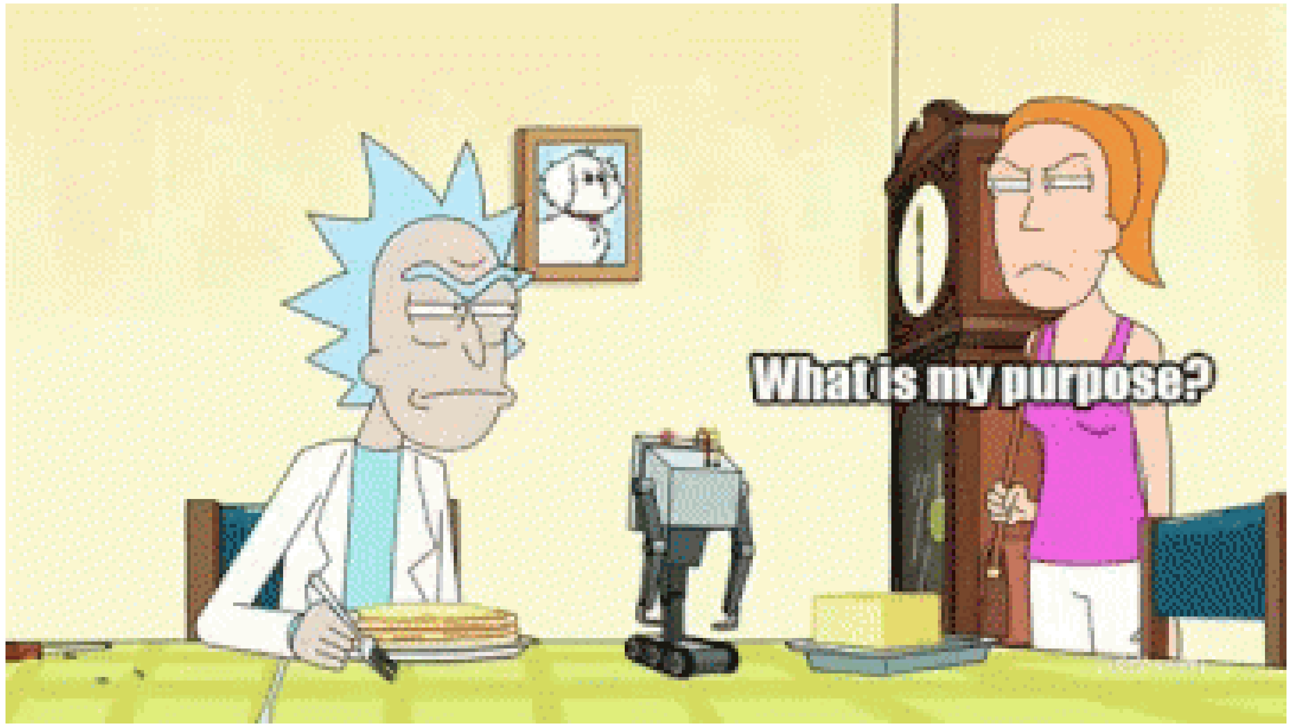
What do Software Engineers actually do?


New Features
Debugging
PR Reviews
Documentations
Proposals
Answering Slack Messages with the most appropriate emoji 🧐


New Features
Debugging
PR Reviews
Documentations
Proposals
Answering Slack Messages with the most appropriate emoji 🧐
Coding
Mental Alignment

Agents can help my team align rapidly

What is an Agent?


What is my purpose?

An agent is an AI system that can:

- Understand a goal or task
- Make decisions about how to accomplish it
- Use tools to interact with the world
- Iterate until the task is complete

Think: **AI + Tools (Fetch, WebSearch, MCP) + Autonomy**

Agents in the Wild

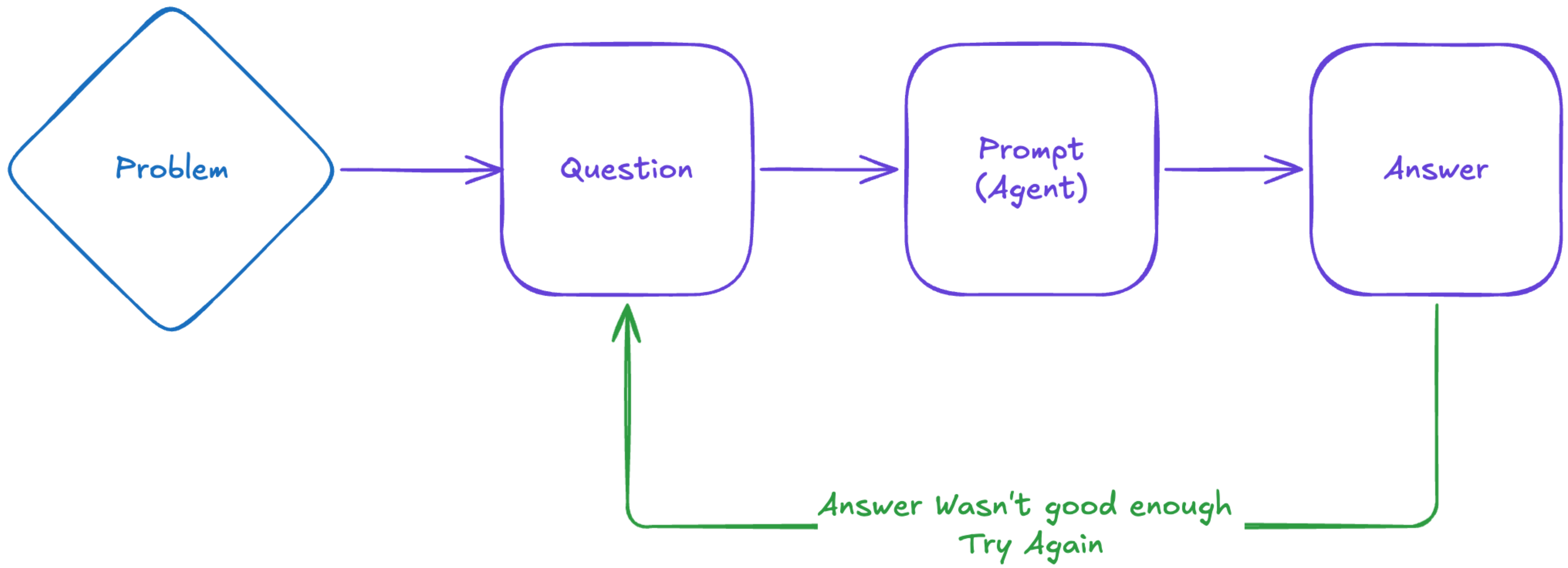
All of these are **single-agent systems**

- **ChatGPT (Web)**: Conversational interface + plugins/actions
- **Claude (Web)**: Conversational interface + plugins/actions
- **Cursor**: Code completion + context-aware edits

All of these are **multi-agent systems**

- **Claude Code**: Long-context reasoning + tool use
- **Cursor (Agent Mode)**: Context-aware edits + tool use

Single-Agent Systems



Query

"Fix this bug for me"

Limited Context

"I don't know how
to answer your question"

"This is mightprobably™
be the answer you're looking for"

Too much Context

"What if you rewrite
React from scratch?"

The Single-Agent Problem

One agent trying to do everything:

- Same reasoning approach for different tasks
- No task-specific optimization

Jack of all trades, master of none

Build your own Custom Agents

Building Custom Agents

Using Claude's `/agent` Command

Create specialized agents with:

- Custom system prompts
- Specific tool sets (Fetch, WebSearch, MCPs)
  - Can also restrict usage
- Defined output formats
- Task-focused behavior

Building Custom Agents

Example: Creating a Specialized Agent

```
/agent
```

Define its purpose:

- You are an expert web developer and markdown specialist with deep expertise in semantic HTML5, CSS frameworks, and data visualization.
- Your primary responsibility is to convert markdown documents into well-structured, accessible, and beautifully styled HTML
- Use the Picnic CSS framework (https://picnicss.com/).

DEMO!

Specialized Agents

Each agent has a **single responsibility**:

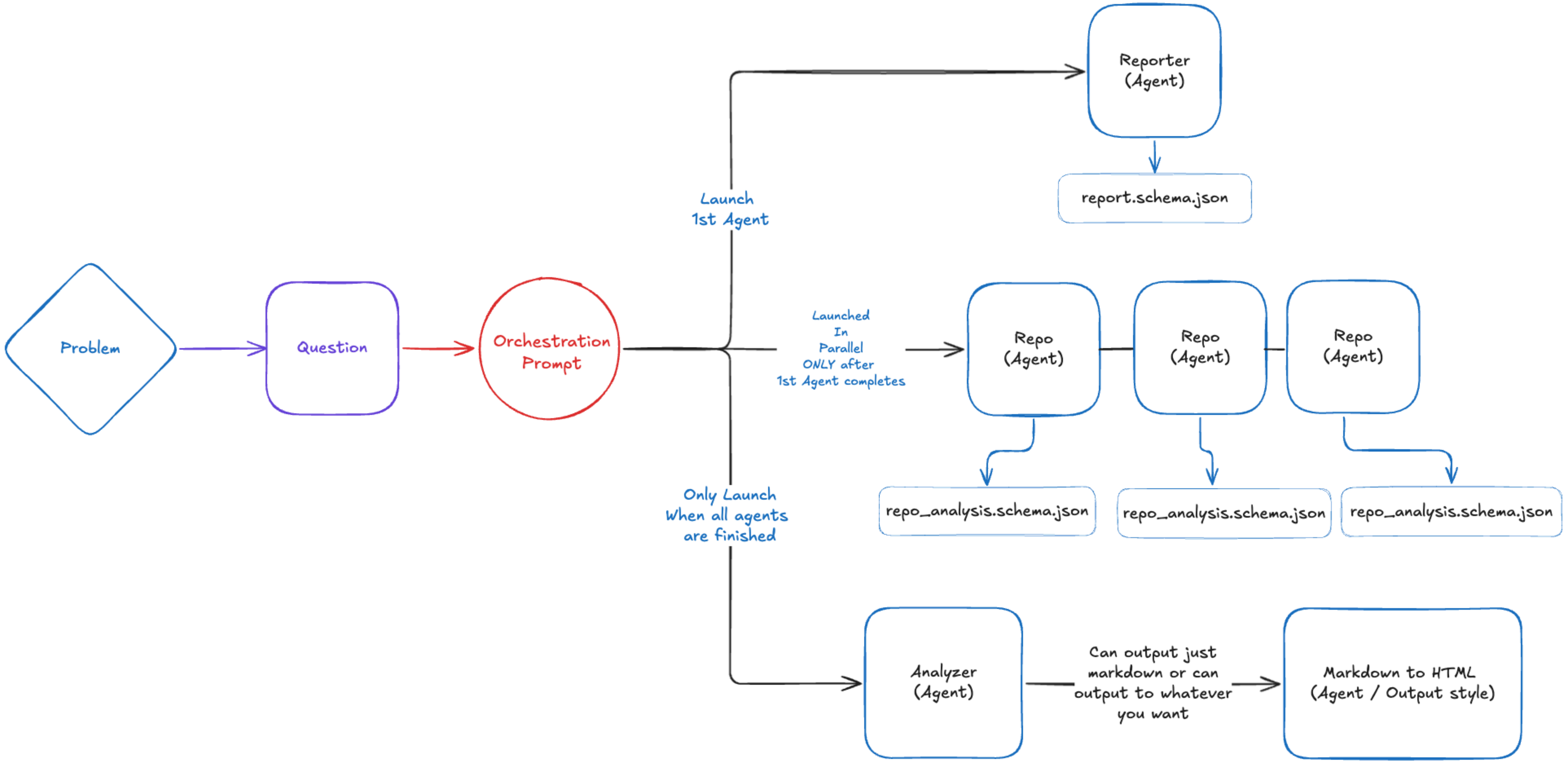
1. **Reporter**: Analyze a GitHub issue deeply and provide comprehensive context about the problem, including identifying related repositories that should be examined for additional context.
2. **Repo Finder**: Given a specific repository and search context, clone it, analyze it for relevant files and patterns, and output your findings.
3. **Analyzer**: Synthesize all context from Agent Reporter and Agent Repo Getter to produce a comprehensive analysis of the GitHub issue. Your output will be the definitive resource for understanding the problem.

Okay, that's cool. But so what?!

Multi-Agent Flow in Action

The Power of Orchestration

Using Claude's `/command` we can build an orchestration prompt that allows us to coordinate how all of these individual agents should work together


Problem
Question
Orchestration Prompt
Launch 1st Agent
Reporter (Agent)
report.schema.json
Launched In Parallel ONLY after 1st Agent completes
Repo (Agent)
Repo (Agent)
Repo (Agent)
repo_analysis.schema.json
repo_analysis.schema.json
repo_analysis.schema.json
Only Launch When all agents are finished
Analyzer (Agent)
Can output just markdown or can output to whatever you want
Markdown to HTML (Agent / Output style)

Claude Commands

Create an Orchestration Prompt this within `./claude/commands`

```
/analyze-issue https://github.com/TanStack/form/issues/1874
```

DEMO #2

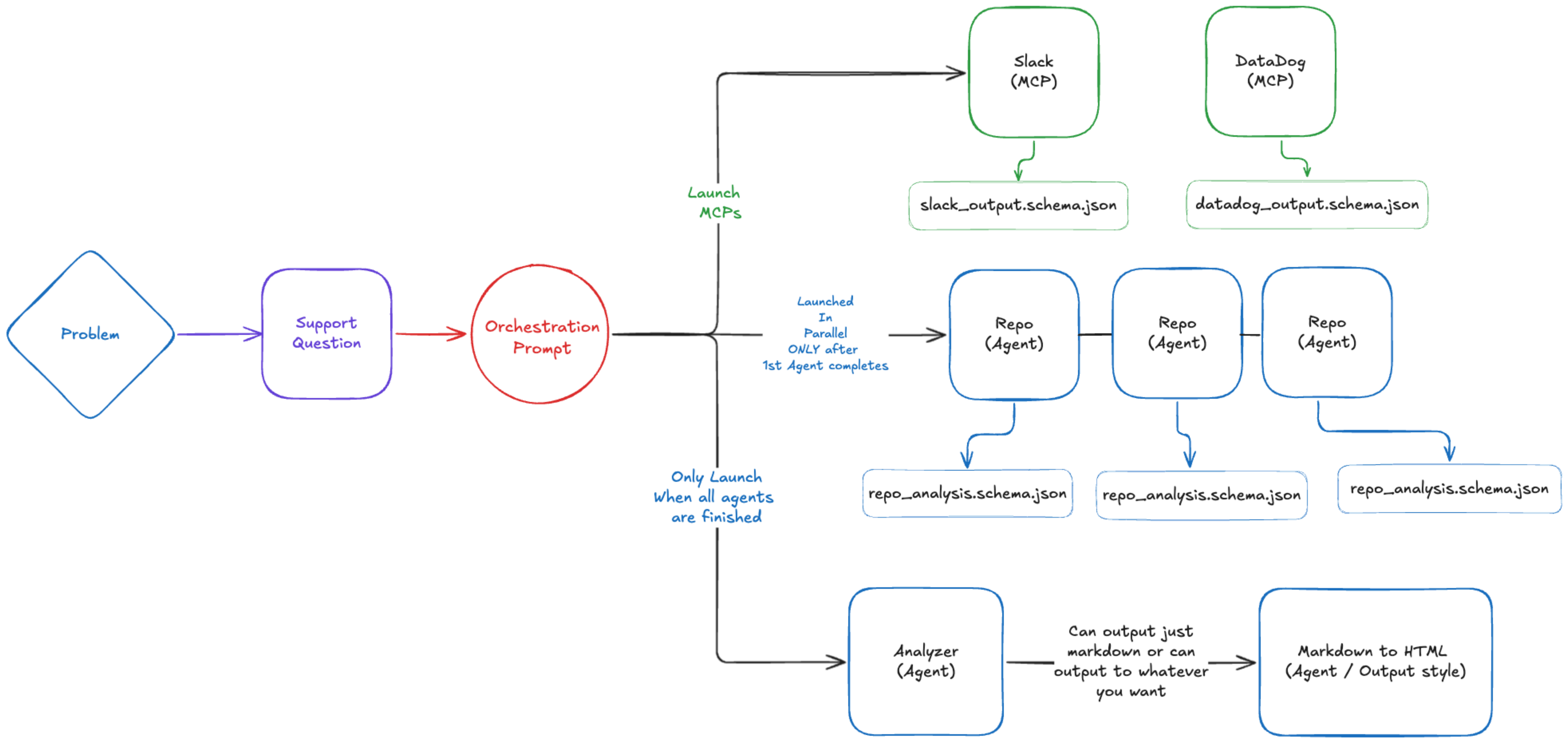
Multi-Agent Flow in Action

Why This Works

Each agent:

- Specializes in one thing
- Produces optimized output for its task
- Passes structured data to the next agent

Result: Better than any single agent could achieve


Problem
Support Question
Orchestration Prompt
Launch MCPs
Slack (MCP)
DataDog (MCP)
slack_output.schema.json
datadog_output.schema.json
Launched In Parallel ONLY after 1st Agent completes
Repo (Agent)
Repo (Agent)
Repo (Agent)
repo_analysis.schema.json
repo_analysis.schema.json
repo_analysis.schema.json
Only Launch when all agents are finished
Analyzer (Agent)
Can output just markdown or can output to whatever you want
Markdown to HTML (Agent / Output style)

Key Takeaways

1. Single agents have inherent limitations
2. Specialization beats generalization
3. Custom agents are easier to build than you think
4. Multi-agent flows unlock new capabilities
5. You can start building today

Questions?

Thank you!