



You wrapped an API in an "agent"

Well done.

// Congratulations, you now have:

- + Added latency
- + Unnecessary complexity
- + Harder debugging

Let's fix that. 🙅



The **Over-Agent** Trap

Teams are creating agents
because it's **hot**
not because it **helps**

"Agentic AI" is the new microservices 😅



3 Questions Before You Agent

1 Does it need autonomous decision-making?
If it's deterministic, use a function.

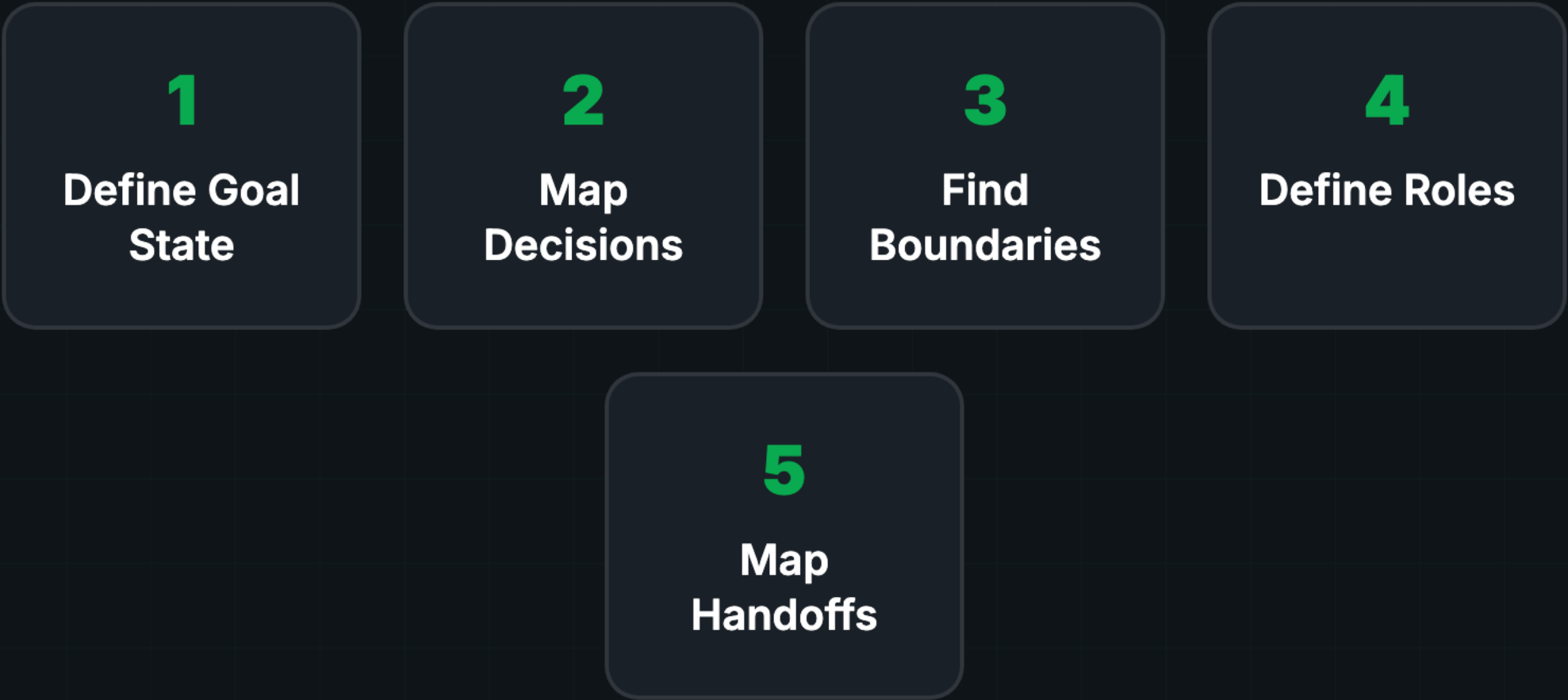
2 Will it benefit from NL reasoning?
Agents shine with ambiguity.

3 Is coordination genuinely complex?
Multiple agents need multiple reasons.

"No" to all 3? Skip the agent.



The 5-Step Decomposition Framework



Goal → Decisions → Boundaries → Responsibilities → Handoffs

The thinking before the diagramming.



Find the Natural Boundaries



Different Domains

SQL expertise vs visualization design = different agents



Different Tools

DB access vs API calls = different capability profiles



Different Oversight

Human review needed vs autonomous = cleaner governance



Different Failures

One failing shouldn't block partial value from others

These signals reveal where to split.



One Sentence Per **Agent**

RESPONSIBILITY STATEMENT

"The **Query Agent** interprets natural language questions and generates appropriate SQL queries against the sales database."

RESPONSIBILITY STATEMENT

"The **Validation Agent** checks query results against business rules and flags anomalies before presenting to users."

Can't write it in one sentence?

Split it further.



The best agentic systems don't have the **most agents**

They have the **right** ones.



Read the full breakdown
myyearindata.com



Follow me on LinkedIn
Scott Bell



DailyDatabricks.Tips



Databricks.News

Powered by 