From Chatbots to Multi-Agent Systems

Lessons from building Agentic RAG in production

Corey Yang-Smith

SOVRA

Table of Contents

SOVRA

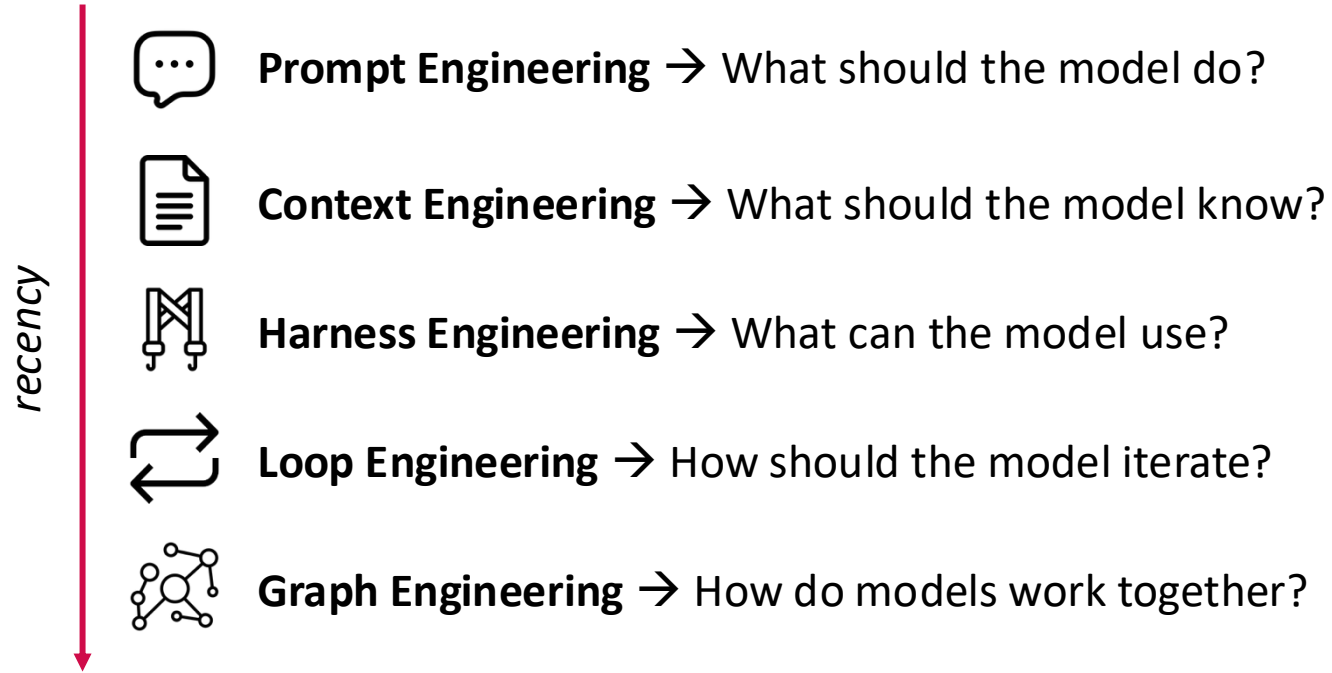
Where Chatbots Break Down

From simple inference to retrieval augmented generation

SOVRA

History of Terminology

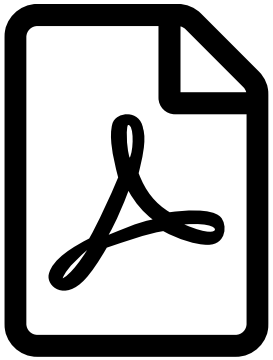
Evolving nomenclature that reflects technological progress



SOVRA

A Good Answer is not a Completed Workflow

An example in retrieval and document QA

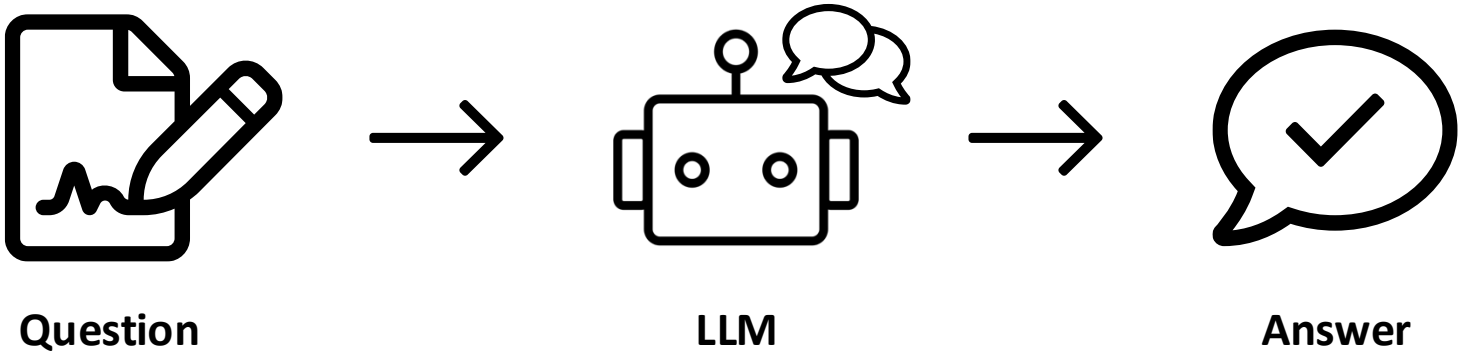


Goal: Given some document(s), how can we answer a user's question?

SOVRA

A Good Answer is not a Completed Workflow

An example in retrieval and document QA – **Simple Inference**



Question **LLM** **Answer**

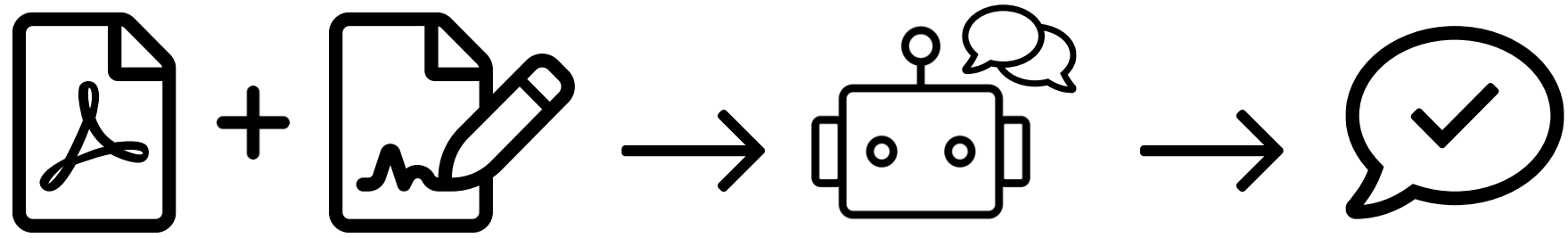
What is the scope of work for this solicitation?

I do not see any solicitation document.

SOVRA

A Good Answer is not a Completed Workflow

An example in retrieval and document QA – **Simple Inference**



Document **Question** **LLM** **Answer**

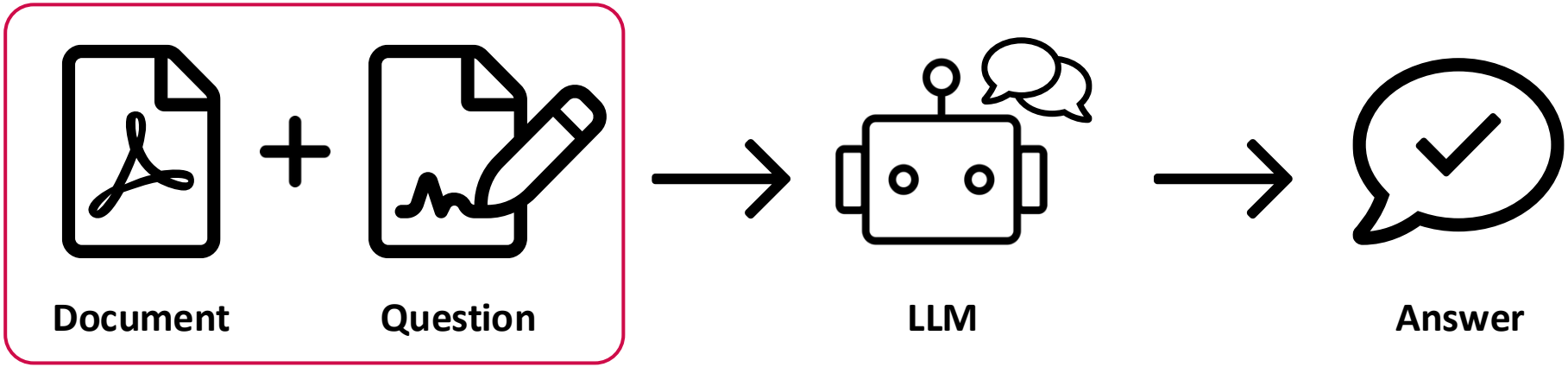
What is the scope of work for this solicitation?

The scope of work is…

SOVRA

A Good Answer is not a Completed Workflow

An example in retrieval and document QA – **Simple Inference**



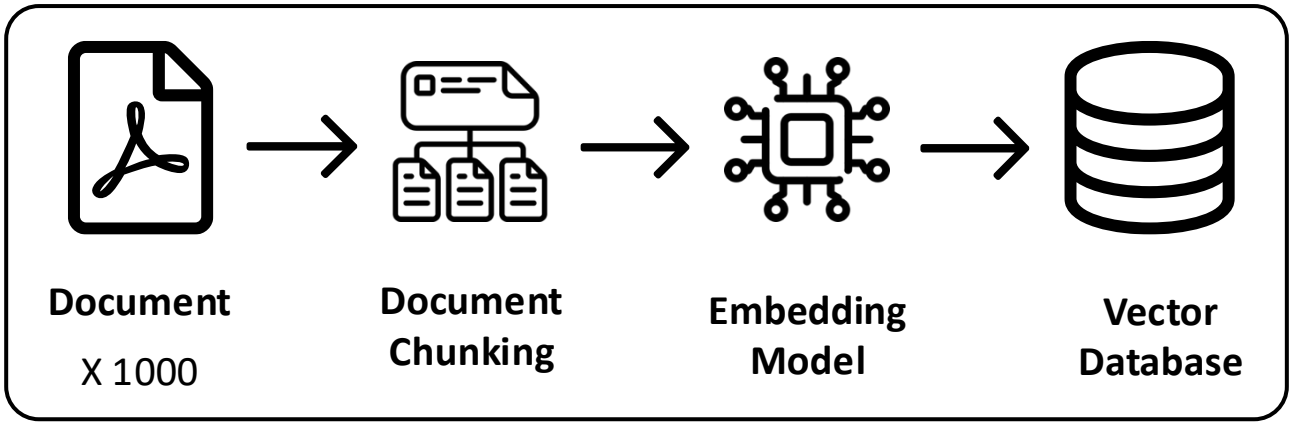
Prompt Engineering

1M Tokens → 750K Words → 3000 pages

SOVRA

A Good Answer is not a Completed Workflow

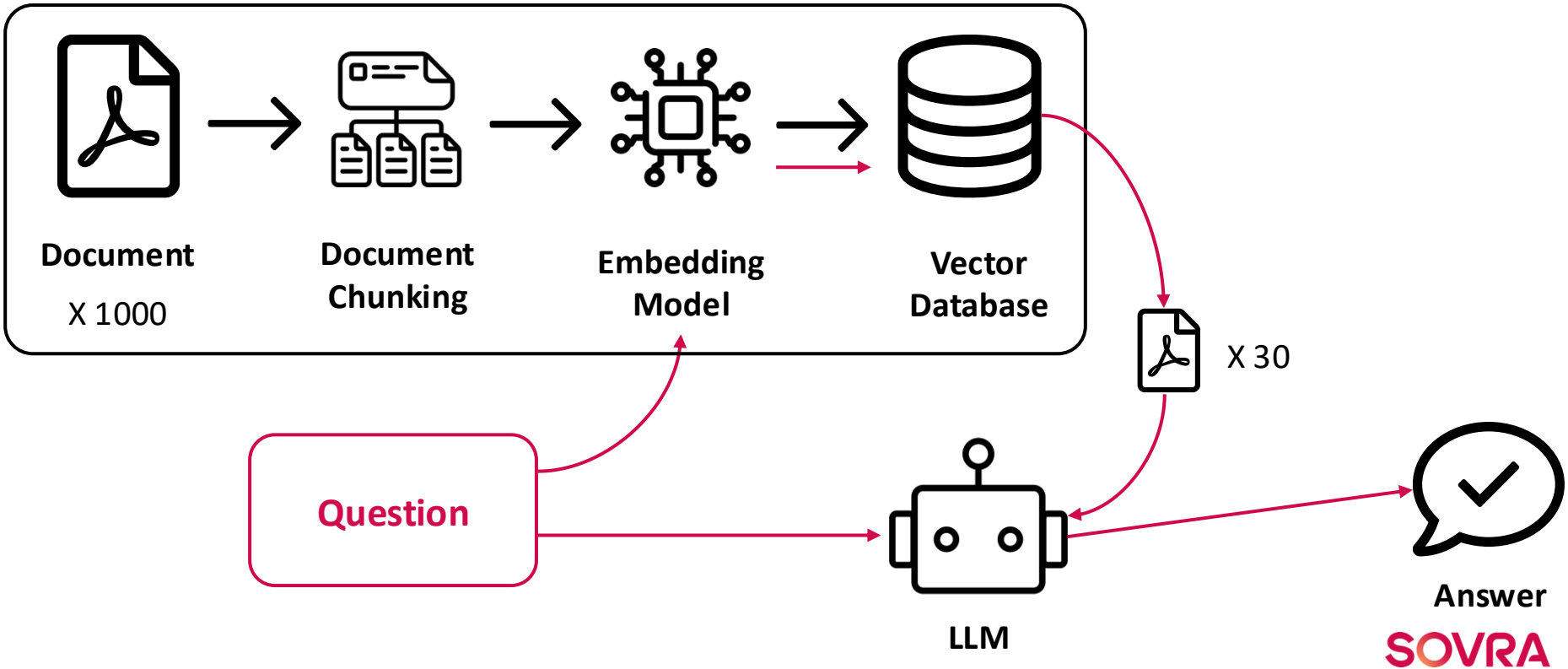
An example in retrieval and document QA – **Simple RAG**



SOVRA

A Good Answer is not a Completed Workflow

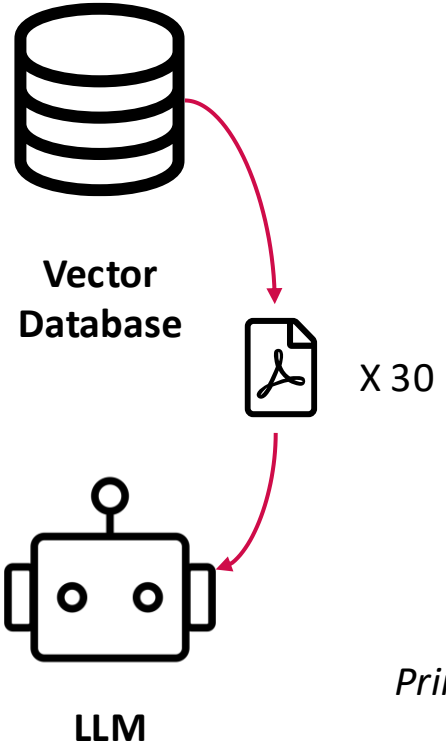
An example in retrieval and document QA – **Simple RAG**

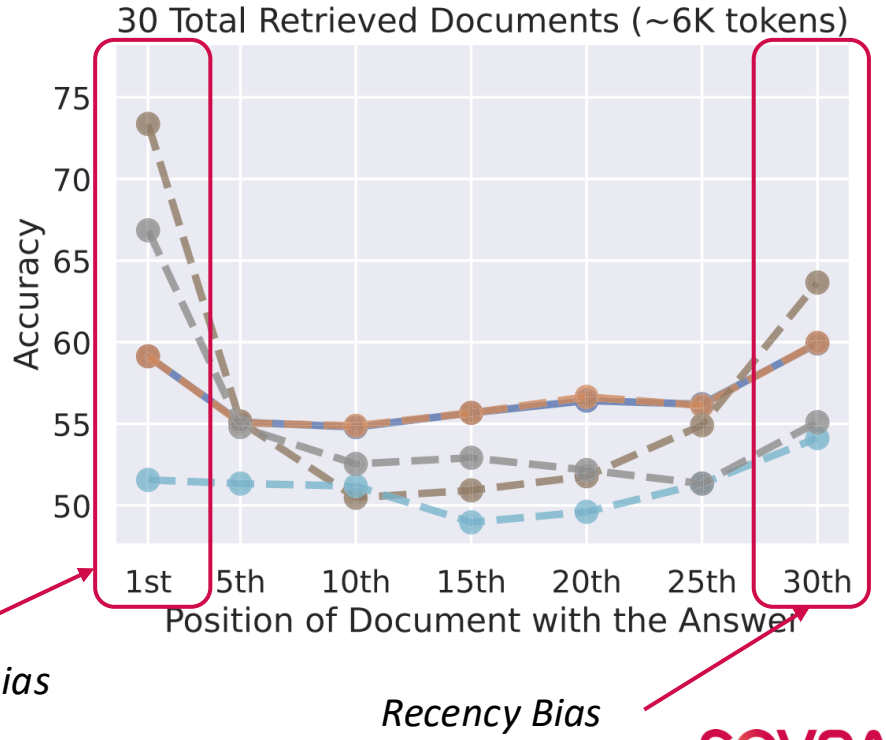


SOVRA

Long Context Limitations and Bias in LLMs

First and last position document bias in retrieval



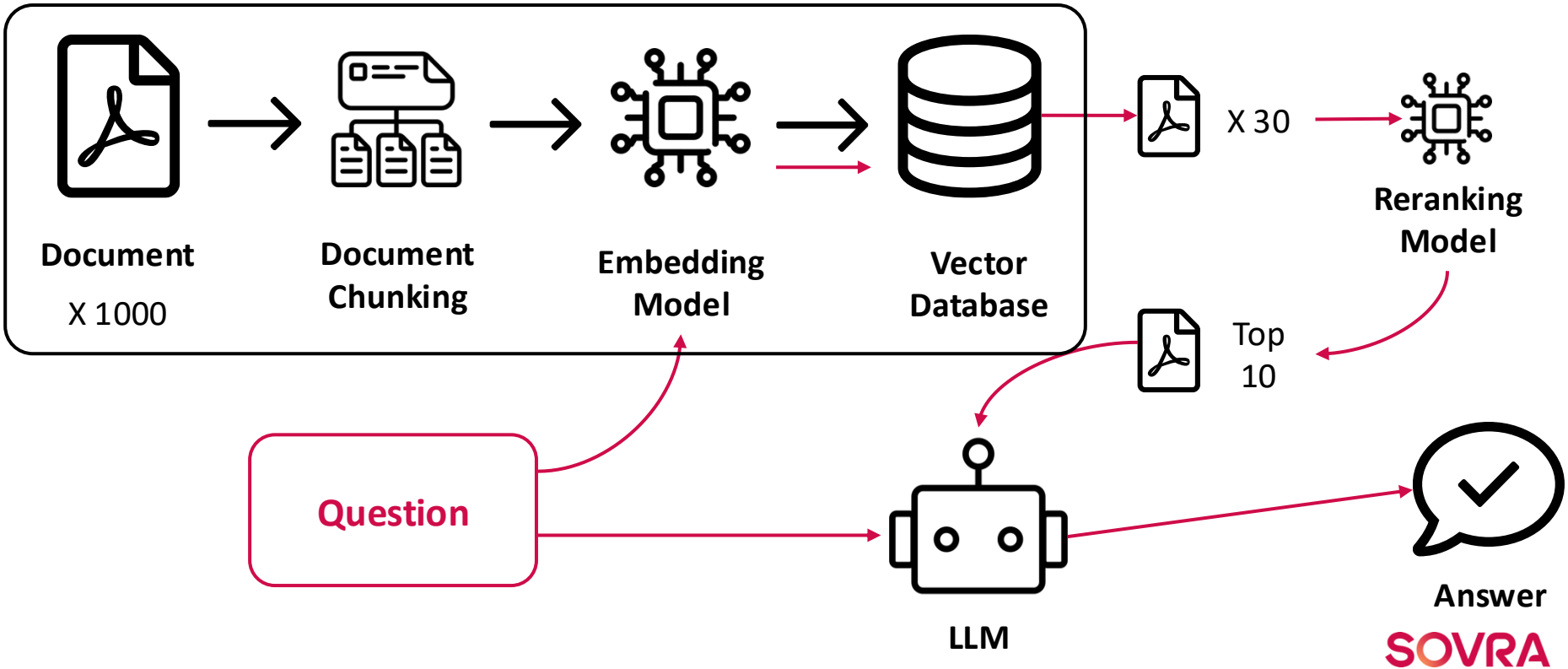


(Liu, 2024) Lost in the Middle: How Language Models Use Long Contexts

SOVRA

A Good Answer is not a Completed Workflow

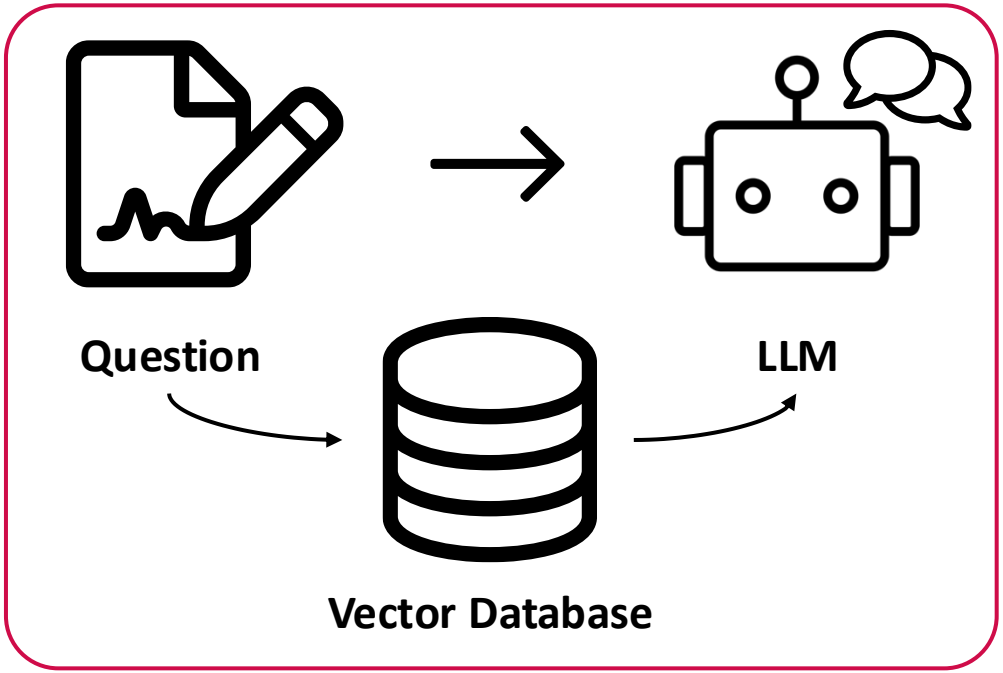
An example in retrieval and document QA – **Simple RAG**



SOVRA

From Model Problem to Systems Problem

Evolving from model challenges to engineering challenges



Context Engineering

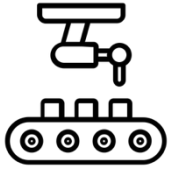
- What chunk size? Are they overlapping?
- Which embedding model?
- How many documents to retrieve?
- Which type of retrieval (lexical, hybrid)?
- Are we rewriting user queries, how?
- Which reranking model are we using?
- Which LLM are we using?
- What prompt are we using?

SOVRA

The Limitations of Chatbot Systems

Trade-offs from strictly engineered systems

Design Decision Overload: Chunking, embeddings, retrieval strategy, top-k, reranking, filtering, prompting, etc



Static Pipeline: Queries vary in complexity, requires sources, and evidence needed, but retrieval needs to be designed in advance.

Retrieval Shortcomings: Evidence can be missed, and noisy evidence can be retrieved – more context does not always improve performance.

SOVRA

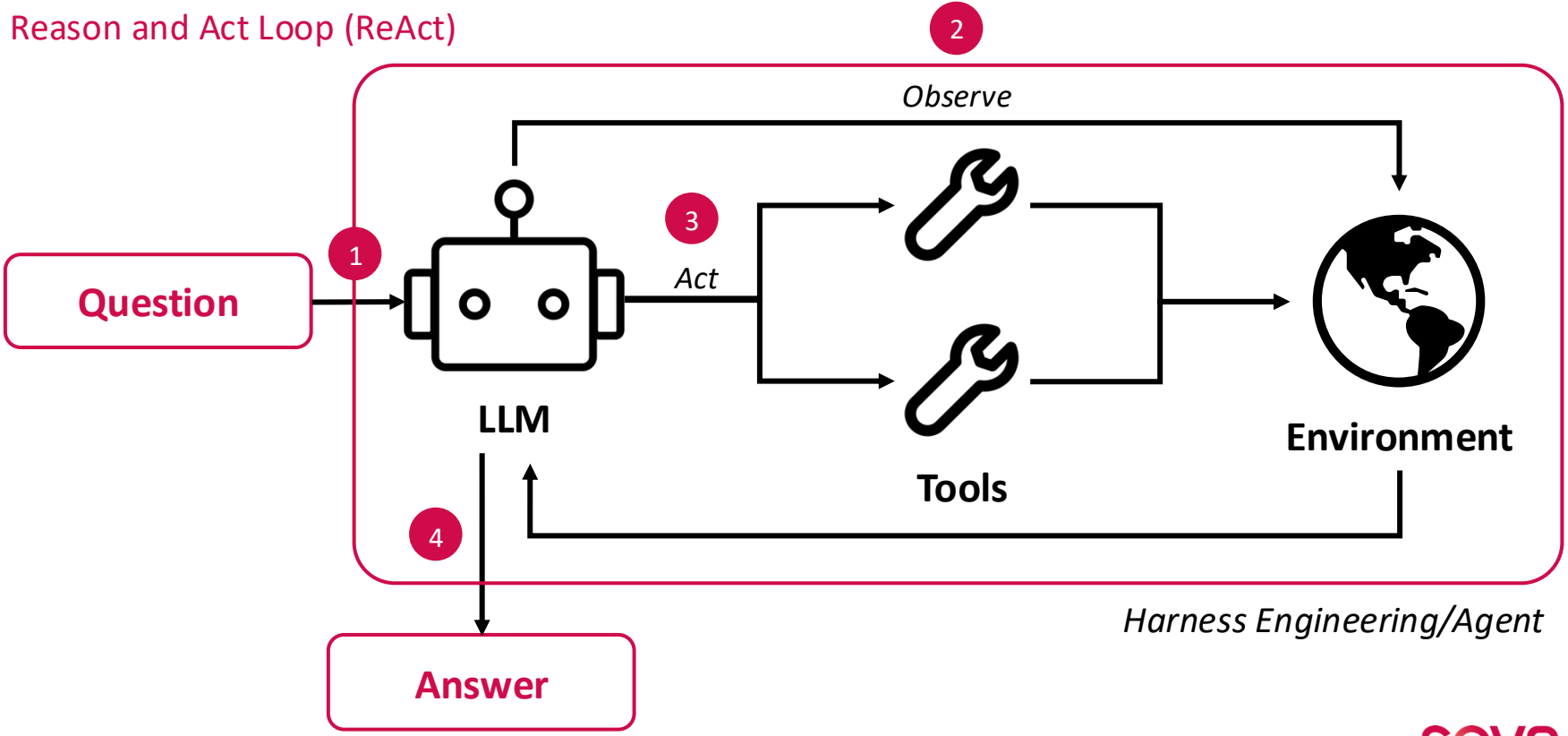
Designing Agentic RAG Workflows

From Retrieval to Iterative Evidence Gathering

SOVRA

What Makes a System Agentic?

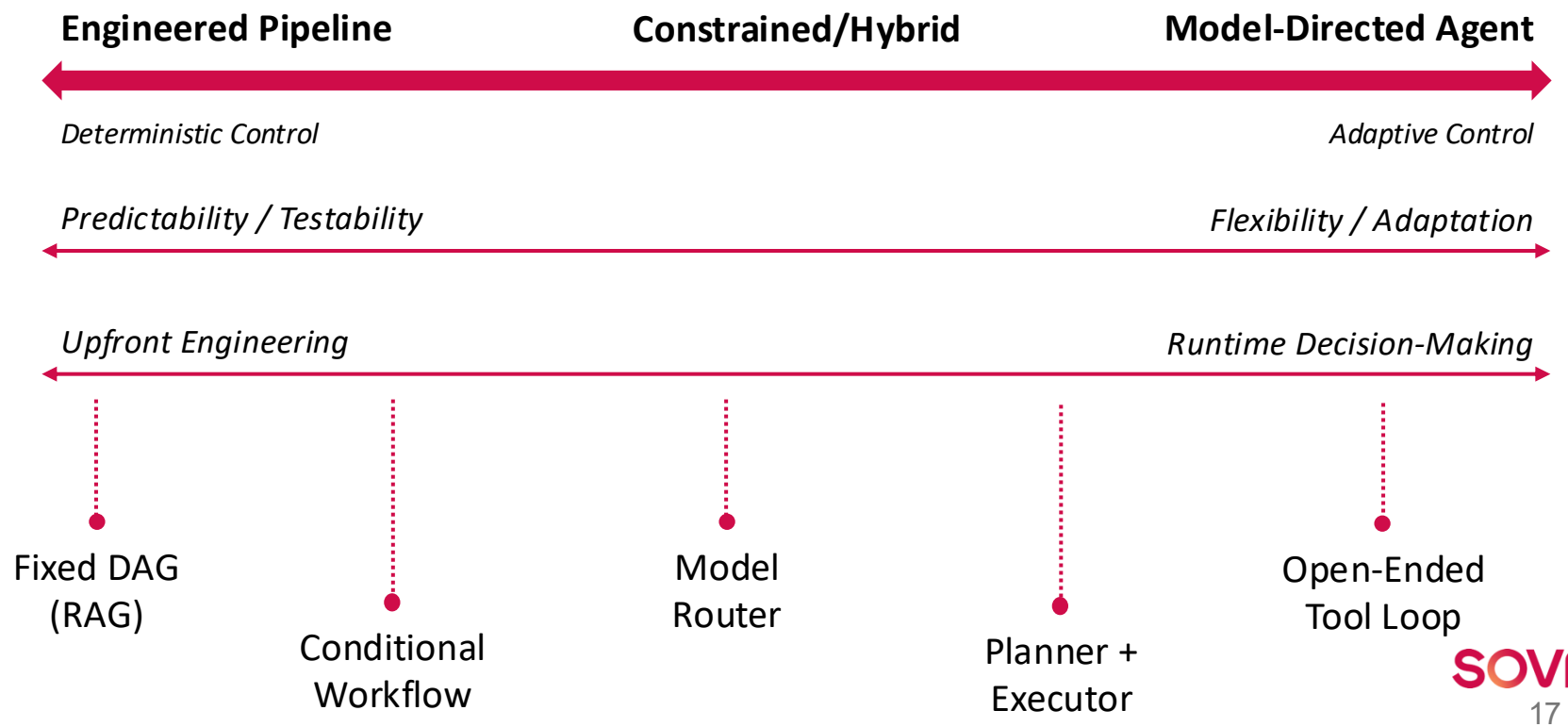
Reason and Act Loop (ReAct)



(Yao, 2022) ReAct: Synergizing Reasoning and Acting in Language Models

Where Should the Model Have Control?

Agentic systems as a spectrum



Agentic AI Design

Design considerations for agentic solutions

Harness Engineering

Defines the "Agent Operating System"

Tools
What can the model do?

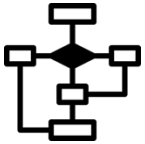
- Search
- Database
- APIs
- Code

State
What can the model observe or remember?

- Evidence
- History
- Scratchpad
- Environment

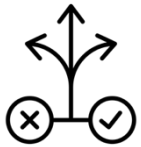
Loop Engineering

Defines the "Agent Control Flow"



Flow
What should happen next?

- Next Action
- Retry/Recovery



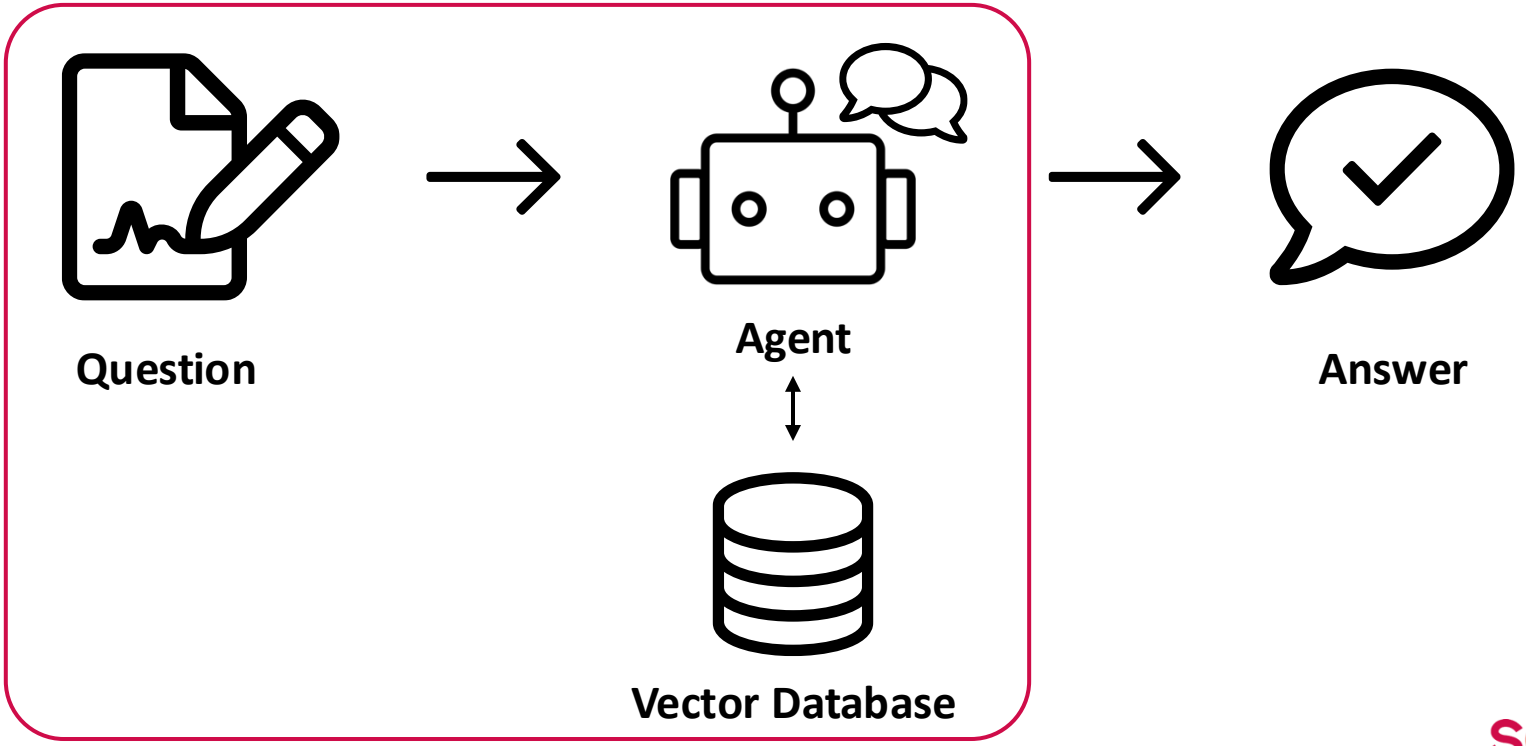
Stopping Criteria
When is the task complete?

- Enough evidence?
- Goal met?
- Max steps/cost?

SOVRA

From Pipeline to Agent

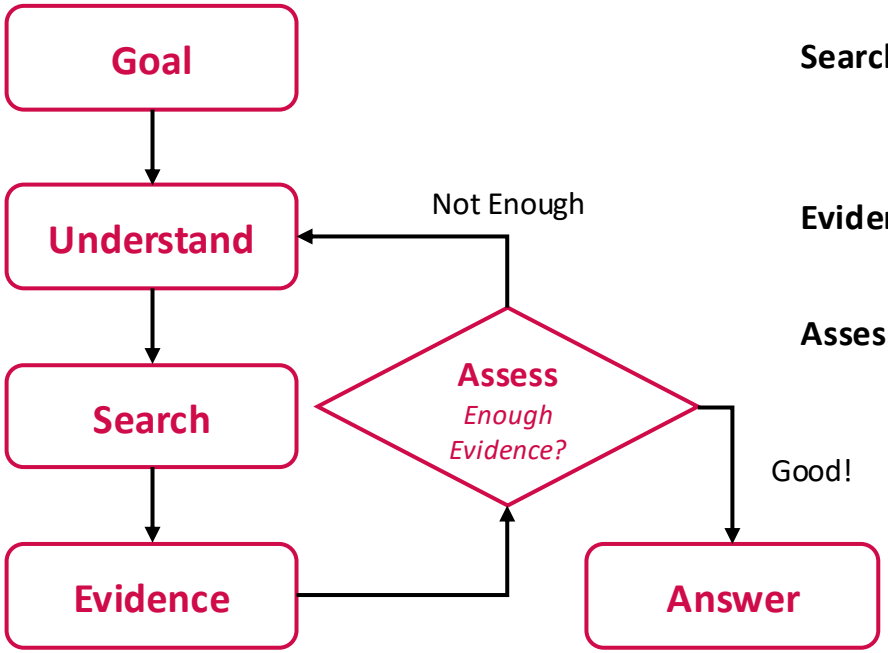
The subtle difference with Agentic RAG



Harness Engineering

Retrieval Becomes Part of the Reasoning Process

Deconstructing the reasoning loop



Understand

What does the user need?
Are there multiple parts to their question?

Search

Which sources do we have available?
How many searches do we need to do?

Evidence

What is specifically relevant to answer the question?

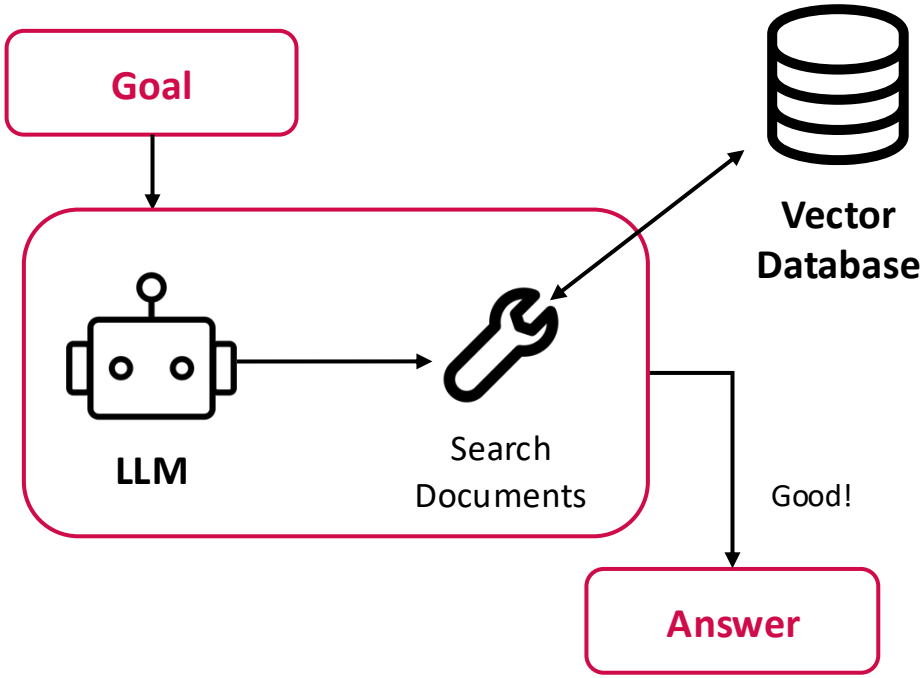
Assess

What is missing?

SOVRA

Retrieval Becomes Part of the Reasoning Process

Deconstructing the reasoning loop



Persona: You are a helpful assistant that answers questions using verified document evidence…

Task/Goal: Answer the user's question…

Tools: You may call `search_documents(query, filters, top_k)` to retrieve sources…

Flow: At each step, decide whether retrieval is necessary, rewrite the query when helpful, choose appropriate search terms…

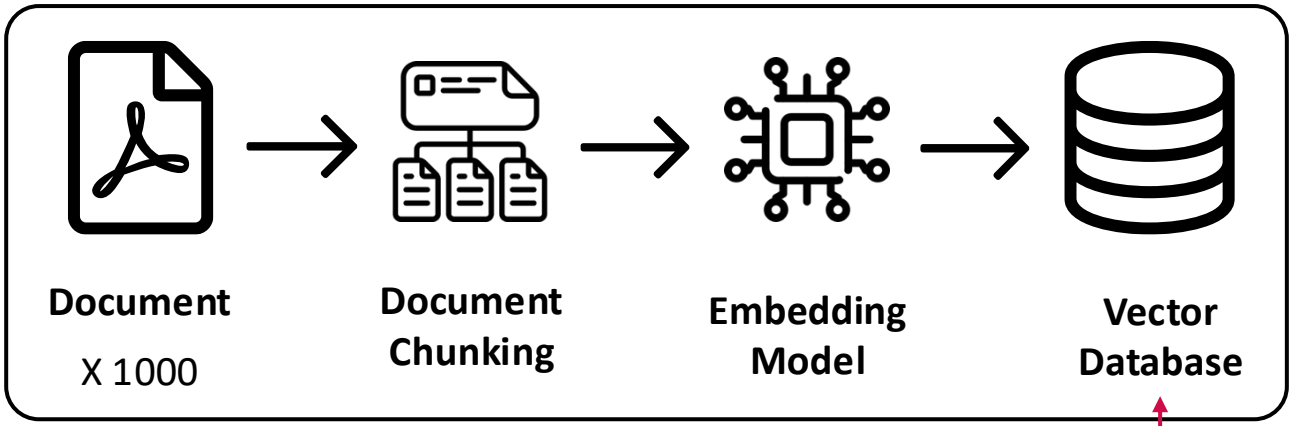
Stopping Criteria: Ask "Do I have enough evidence to answer?" and continue retrieving until the answer is yes…

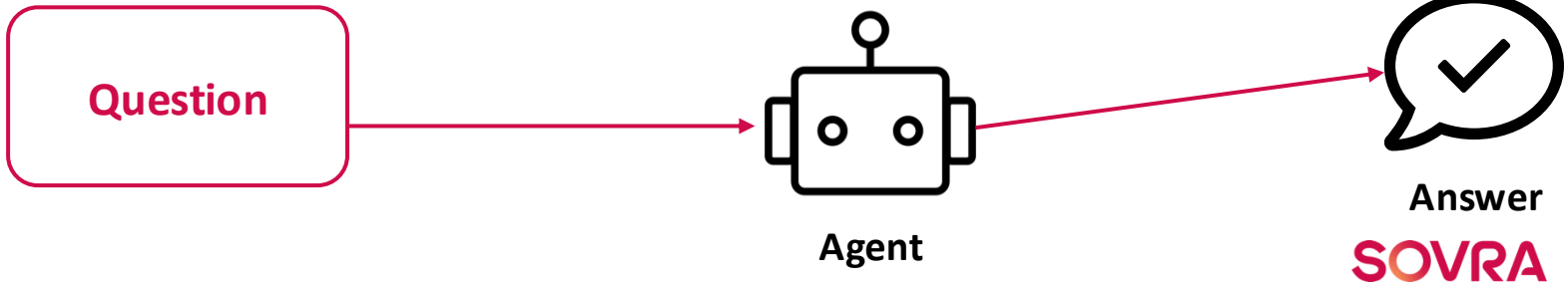
Output: Synthesize the evidence into a concise answer, provide citations…

SOVRA

From Pipeline to Agent

Trading complexity and control





SOVRA

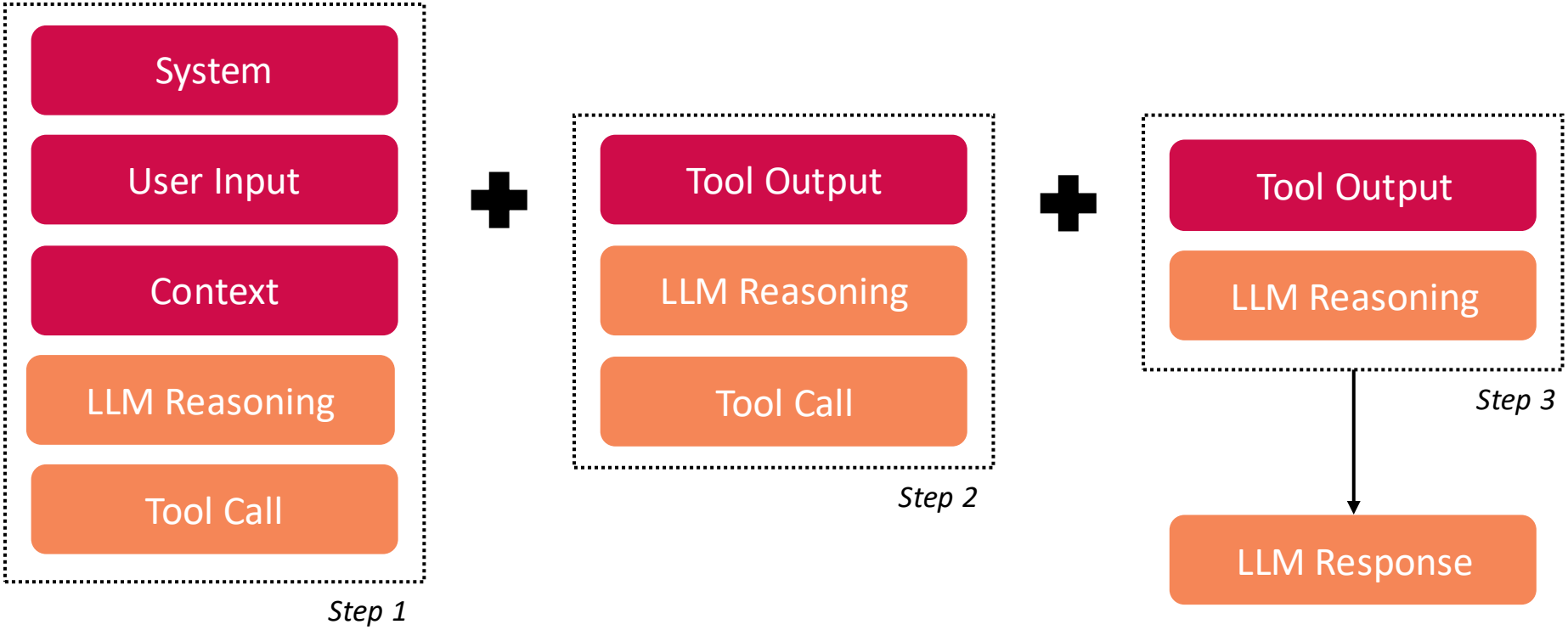
From Design-Time Rules to Runtime Decisions

Retrieval decisions become agent-driven

	Engineered RAG	Agentic RAG
Retrieval	Retrieve top 30 documents	Decide what/whether to retrieve
Source Selection	Always use vector search	Select the appropriate search
Query Strategy	Use a fixed query	Rewrite/decompose query
Search Process	One retrieval pass	Search iteratively
Evidence Evaluation	Fixed cutoff / top-K	Assess if evidence is sufficient
Termination	Always produce an answer	Continue, answer, or stop

SOVRA

Context Is a Managed Resource



Challenges with Single Agent Systems

Trade-offs with single-agent systems

Context Bloat: One agent accumulates all history, evidence, tool outputs, and intermediate states. Higher costs, diluted context, need for summarizations

Serial Execution: Tasks are ordered sequentially, reasoning and tool steps increase latency and extend the critical path

Responsibility Overload: One agent is loosely responsible for many steps, resulting in less specialization,

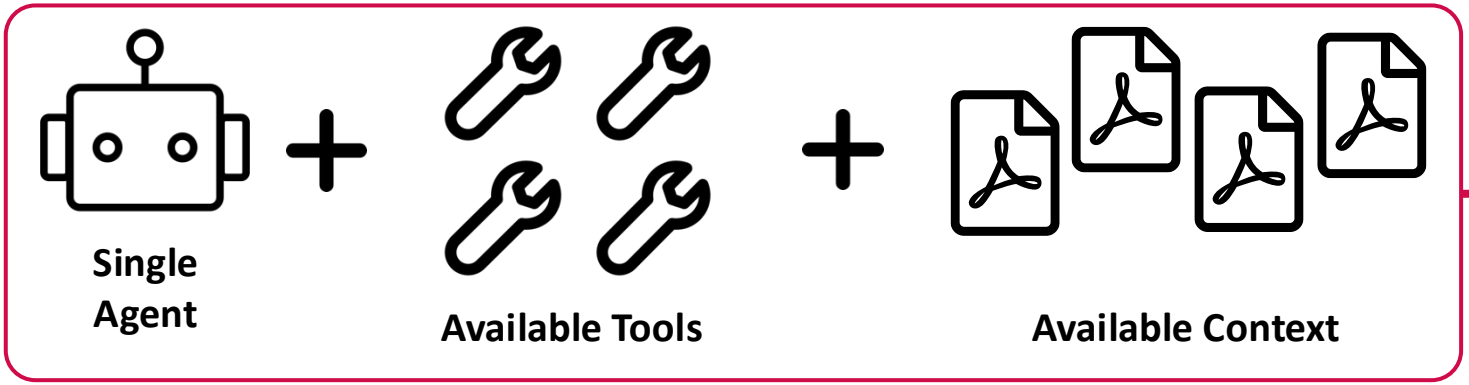
Multi-Agent System Design Patterns

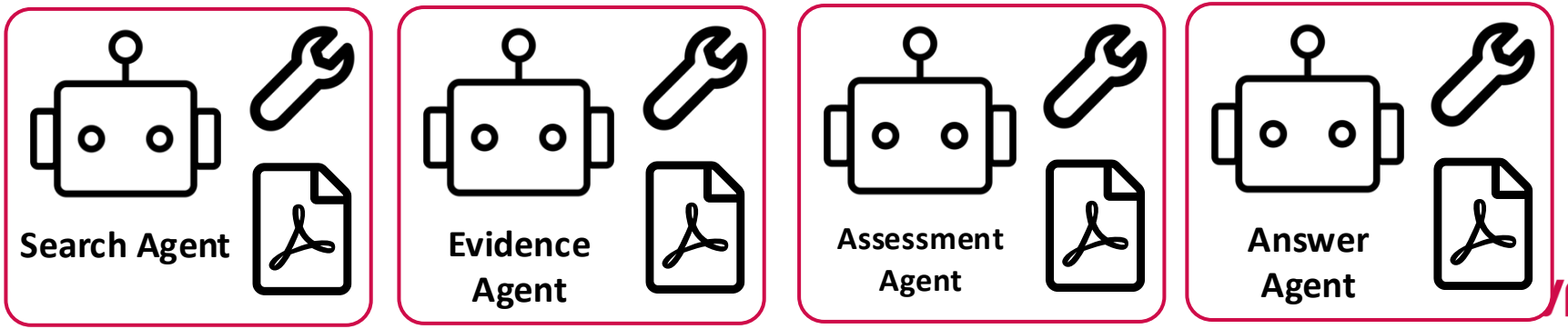
Decomposing Control, Context, and Responsibility

SOVRA

One Agent or Many?

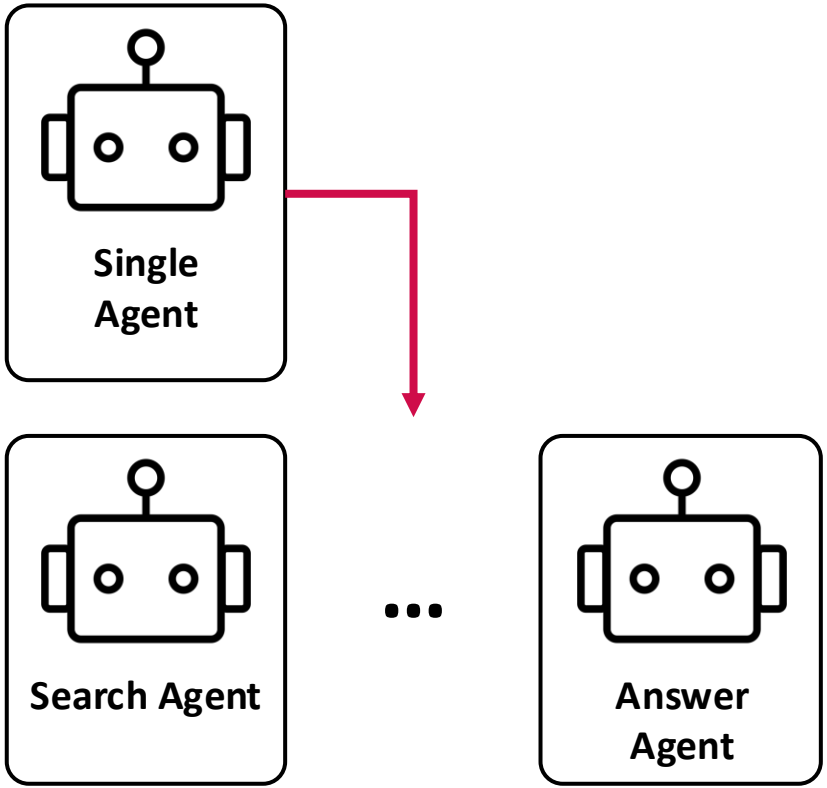
Trading local simplicity for global complexity





One Agent or Many?

Trading local simplicity for global complexity



In moving from single to multi-agent systems, **we gain**:

- Context isolation
- Specialization
- Separation of Concerns
- *Parallelism

We introduce:

- Agent-to-agent handoffs
- Coordination
- Latency/Cost
- More failure modes

SOVRA

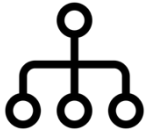
Designing Multi Agent Systems

Four questions that define how agents work together

Orchestration

Who decides what happens next?

Decomposition



How is work divided?

Specialization

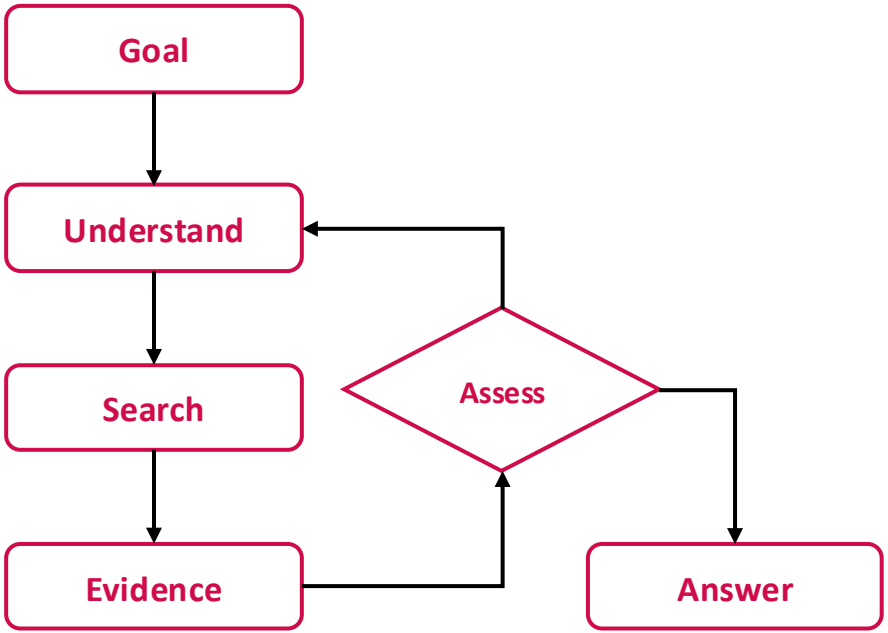
Who performs each task?

Coordination

How are results shared/combined?

Sequential Workflow (Pipeline)

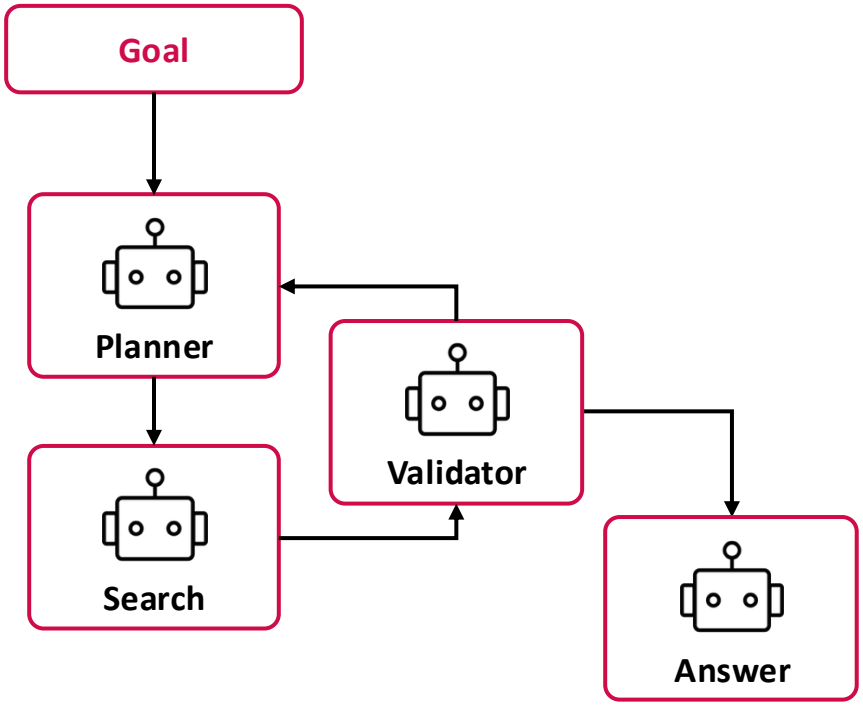
Control flow for agents



SOVRA

Sequential Workflow (Pipeline)

Control flow for agents



Orchestration: Control Flow

Decomposition: Engineered

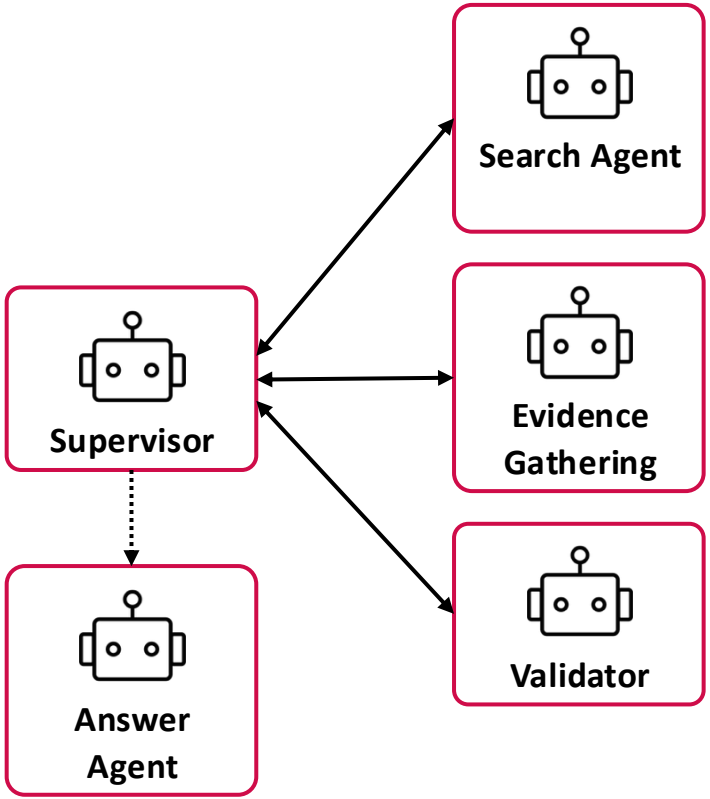
Specialization: Agent/Stage

Coordination: Structured Handoff

SOVRA

Supervisor + Workers

Delegating distributed tasks



Orchestration: Dynamic

Decomposition: Partially Dynamic

Specialization: Workers Expose

Coordination: Supervisor-Driven

SOVRA

Production Trade-offs and Failure Modes

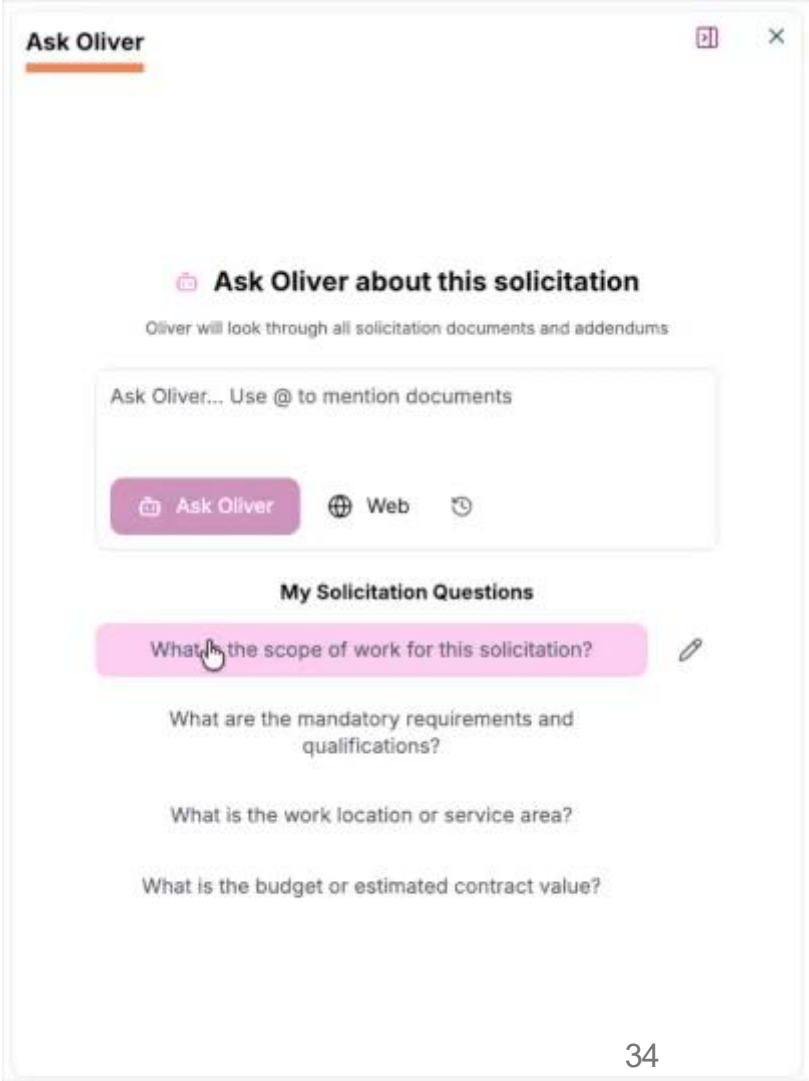
The Hard Part Moves Outside the Model

SOVRA

Who is Oliver?

Your Government Procurement AI Copilot

- **Oliver** is a collection of AI-enabled systems supporting buyers and suppliers in government procurement.
- **Ask Oliver** is our chat service that will retrieve, reason about, and answer user's questions in the context of government documents.



What Gets Harder as the System Becomes (Multi-)Agentic?

Capabilities increases the failure surface

Engineering Complexity

Partial failures become normal. *More state, dependencies, failure modes, and scale to manage.*

Testing & Observability

A bad output doesn't tell you where the failure occurred. *Debugging complexity and failure surface grows.*

Messy Data

Agents cannot reason their way around missing evidence. *Output quality is bounded by the quality of retrieval and evidence.*

Cost, Latency, Reliability

Every loop and handoff extends the critical path. *Every added step trades simplicity for capability, cost, and latency.*

Design for Failure, Not Just the Happy Path

Production agent systems need explicit operational boundaries

1. Engineering Complexity → Make State Explicit

Persist intermediate results, define clear structured handoffs where possible, make steps robust and handle errors, partial completions, etc

2. Testing & Observability → Test each component

Trace every model/tool call, evaluate end-to-end and individual components

3. Messy Data → Treat content as evidence

Preserve citations, validate evidence and abstain answering when insufficient

4. Cost, Latency, Reliability → Budget the workflow

Route cheaper models, parallelize work, bound iterations/retries, provide fallbacks.

SOVRA

When is Multi-Agent Worth It?

When Complexity Pays for Itself

SOVRA

When Is Multi-Agent Worth It?

Why Split?

- **Task Decomposition**. Subtasks have clear inputs and outputs and can succeed somewhat independently.
- **Specialized Roles.** Different tasks need different contexts, tools, instructions. Specialization keeps agents focused.
- **Verification.** Separate agents can review, critique, or judge intermediate results before propagating downstream.
- **Parallelism.** Independent subtasks can run concurrently, reducing latency.

What You Pay

- **Lossy Handoffs.** Context, intent, and nuance can be lost as work moves between agents.
- **Coordination Overhead.** Routing, synchronization, combining partial results, become expected engineering problems.
- **Failure Propagation.** Hallucinations or poor outputs propagate downstream.
- **Operational Overhead.** More agents means more state, model calls, tracing, exception handling, retries, latency, and cost to manage.

Default to one agent. Split at clear, observable boundaries when separation provides added value.

SOVRA

Do you have any questions?

Linkedin

SOVRA