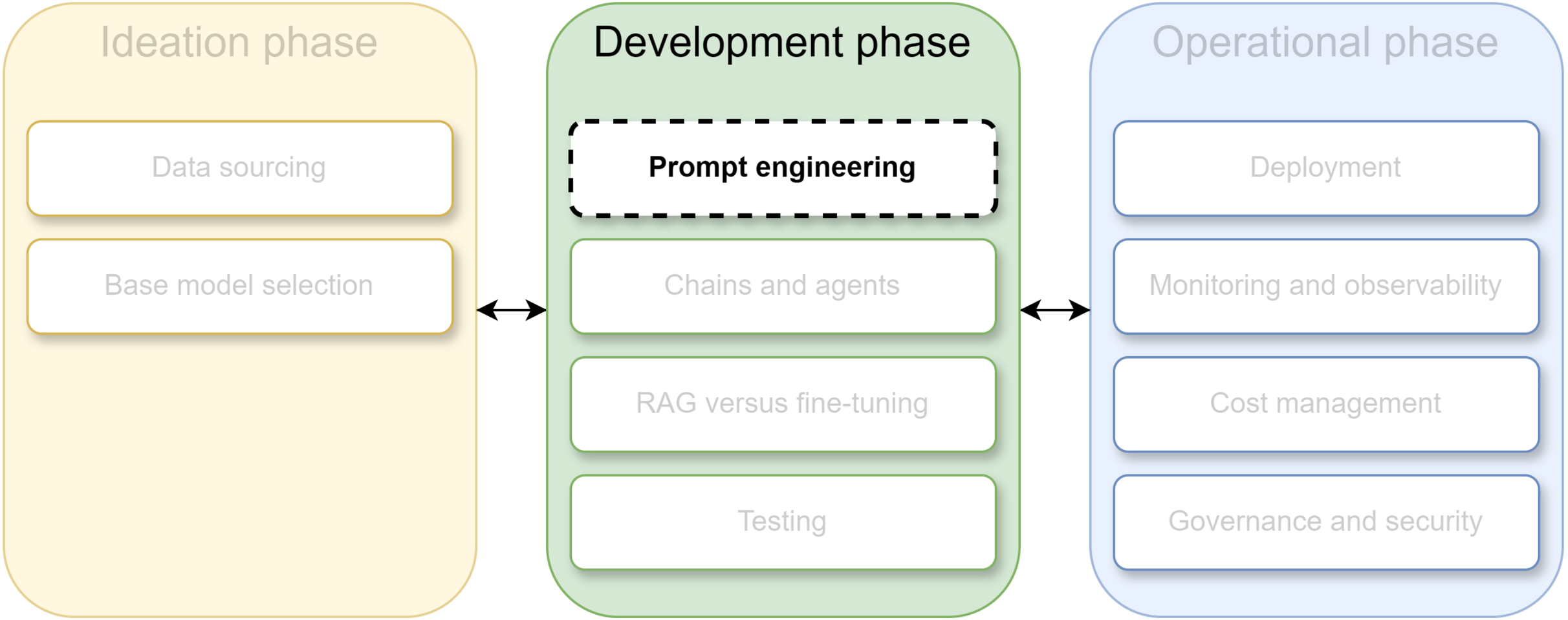
Prompt engineering

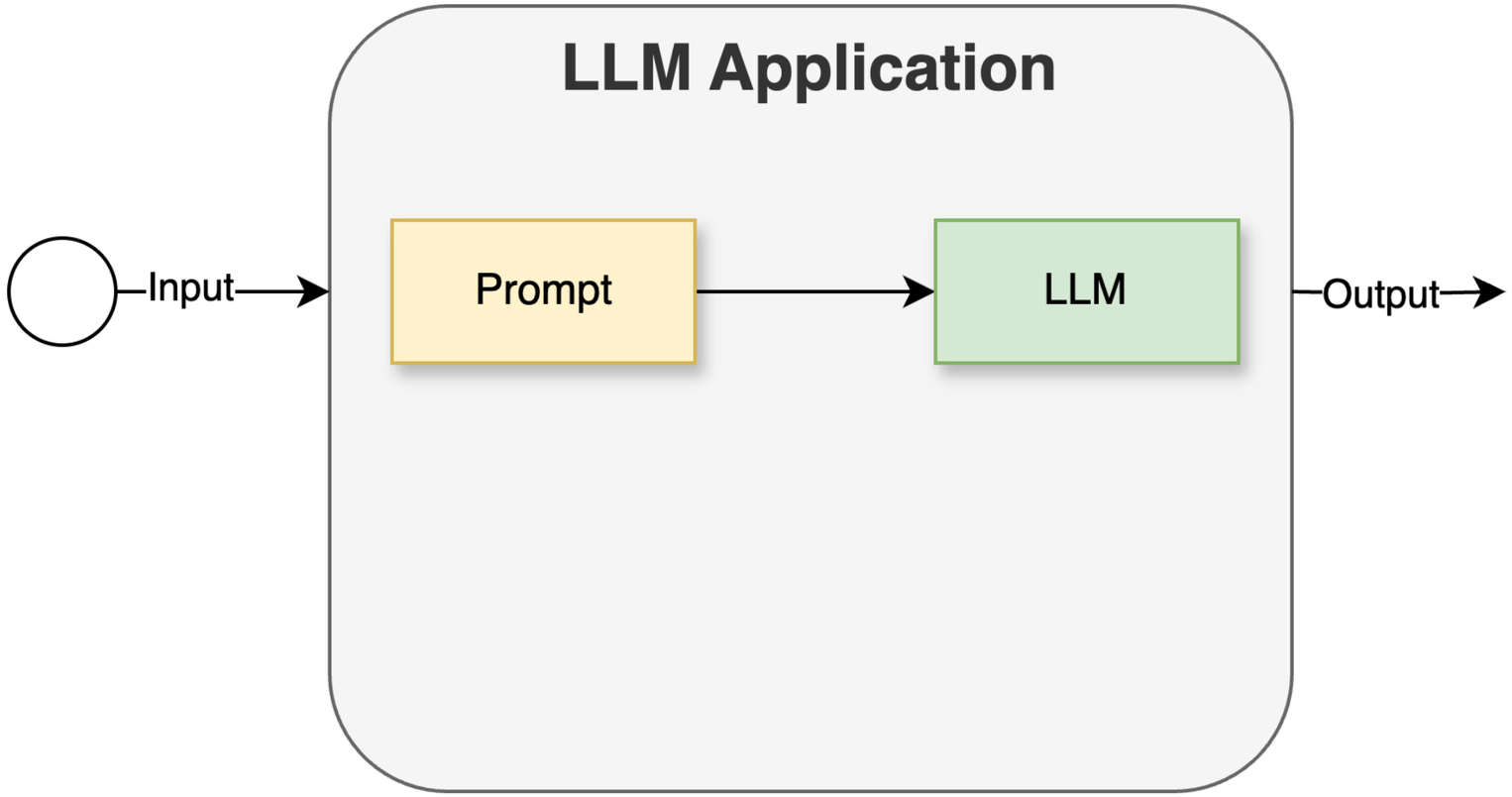
LLMOPS CONCEPTS

Max Knobbout, PhD
Applied Scientist, Uber

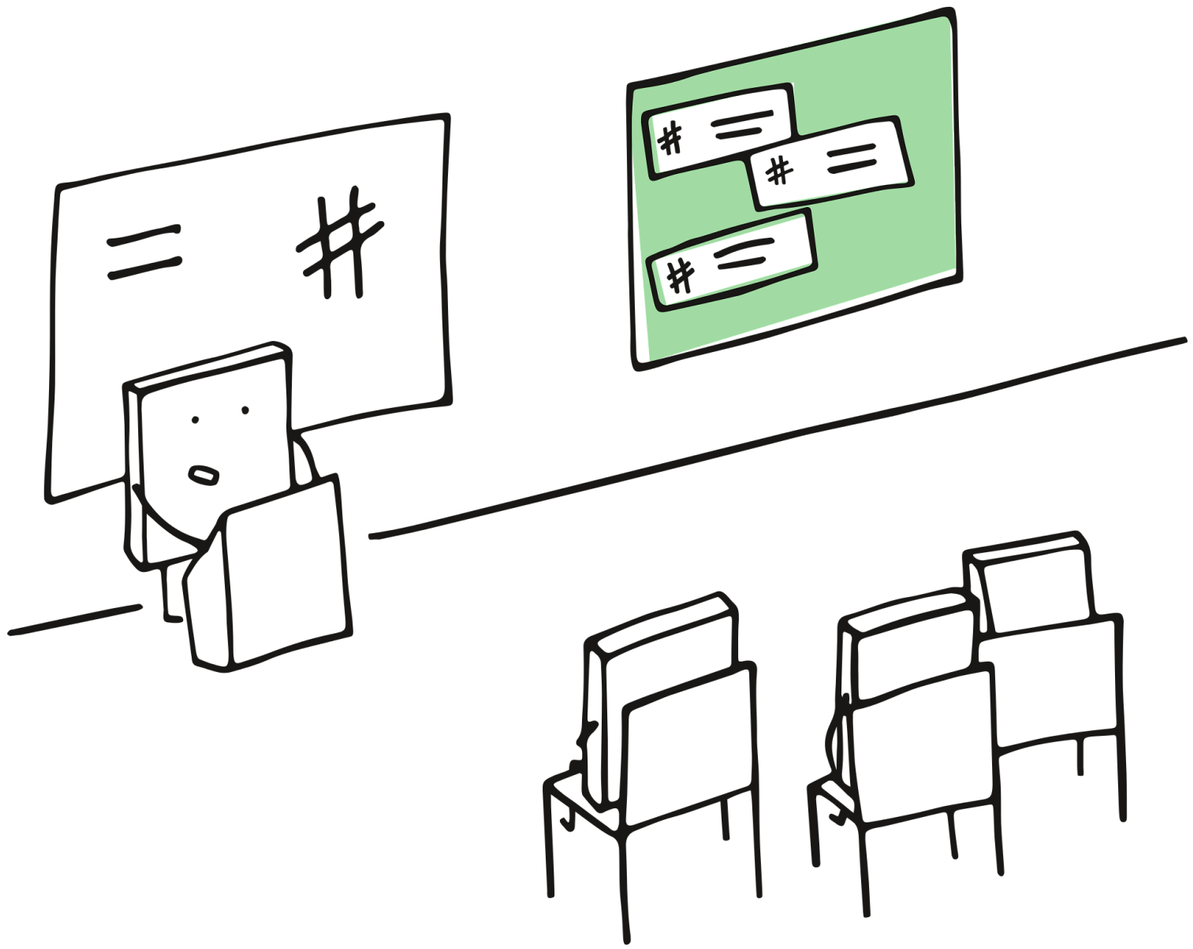
LLM lifecyle: Prompt engineering



The development cycle



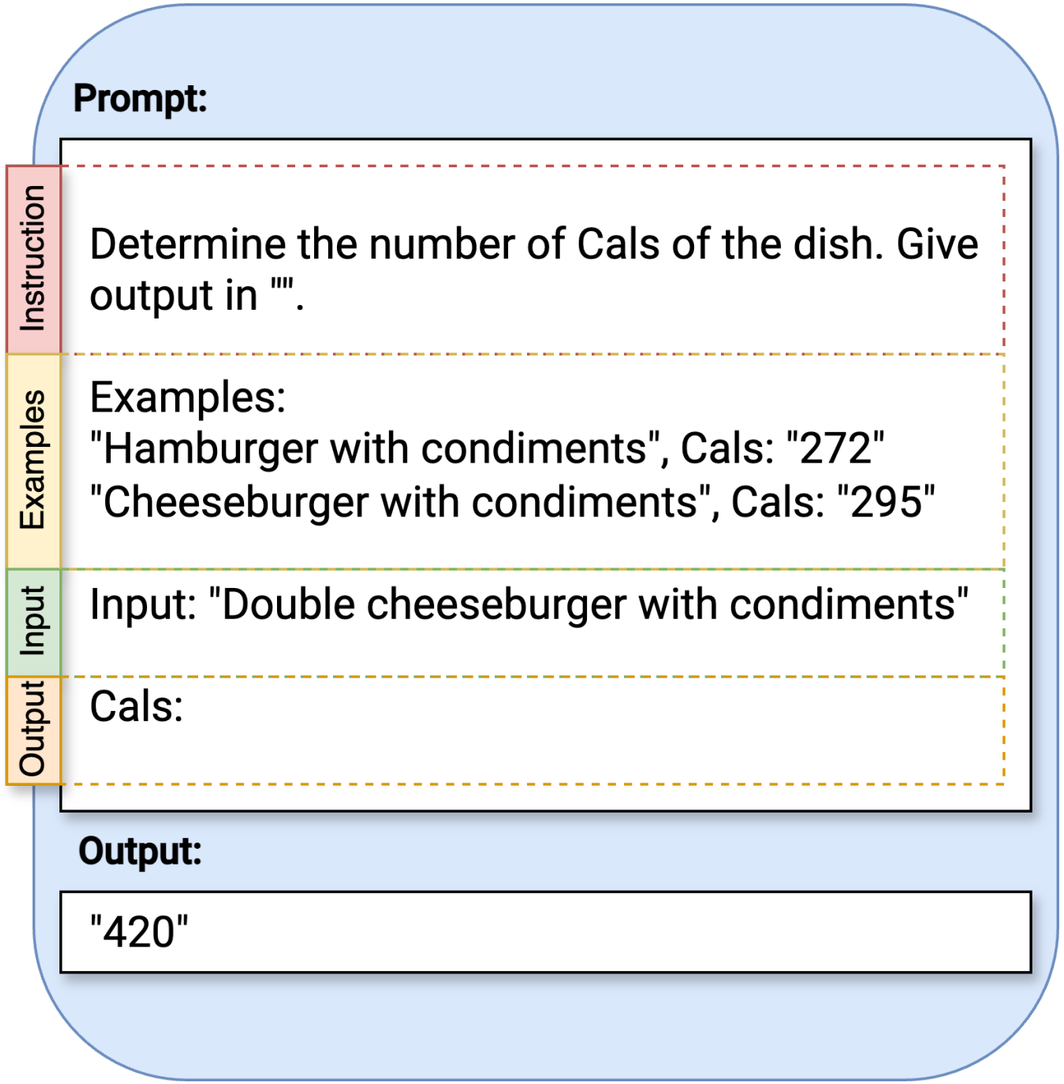
Why is prompt engineering important?



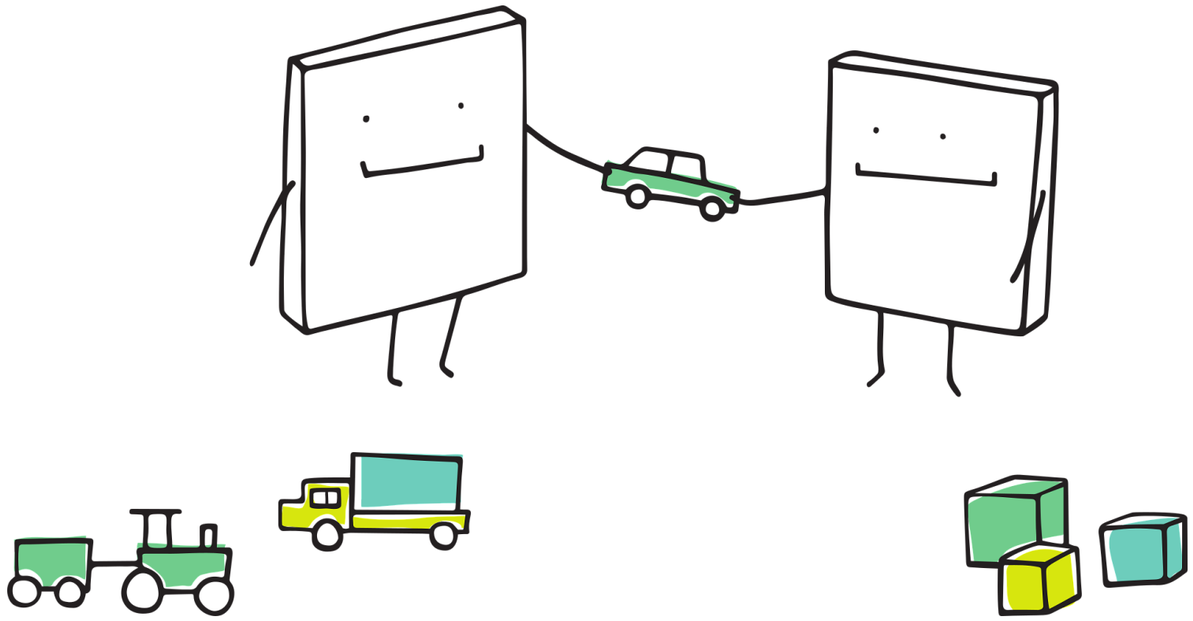
- Improve performance
- Control over output
- Avoid bias and hallucinations

Elements of a prompt

1. Instruction
2. Examples / Context
3. Input data
4. Output indicator



Finding the perfect prompt



Experiment with:

- LLM settings, such as temperature, or max tokens
- In-context learning and other prompt design patterns

Use a playground for experimentation!

Prompt management

- Crucial for efficiency, reproducibility, and collaboration
- Important to track:
  - Prompt
  - Output
  - Model and settings
- Use prompt manager or version control
- Begin generating a collection of good input-output pairs for evaluation

Prompt templates for reusability

Prompt

Determine the number of Cals of the dish.
Give output in "".

Examples:
"Hamburger with condiments", Cals: "272"
"Cheeseburger with condiments", Cals: "295"

Input: "Double cheeseburger with condiments"

Cals:

Template

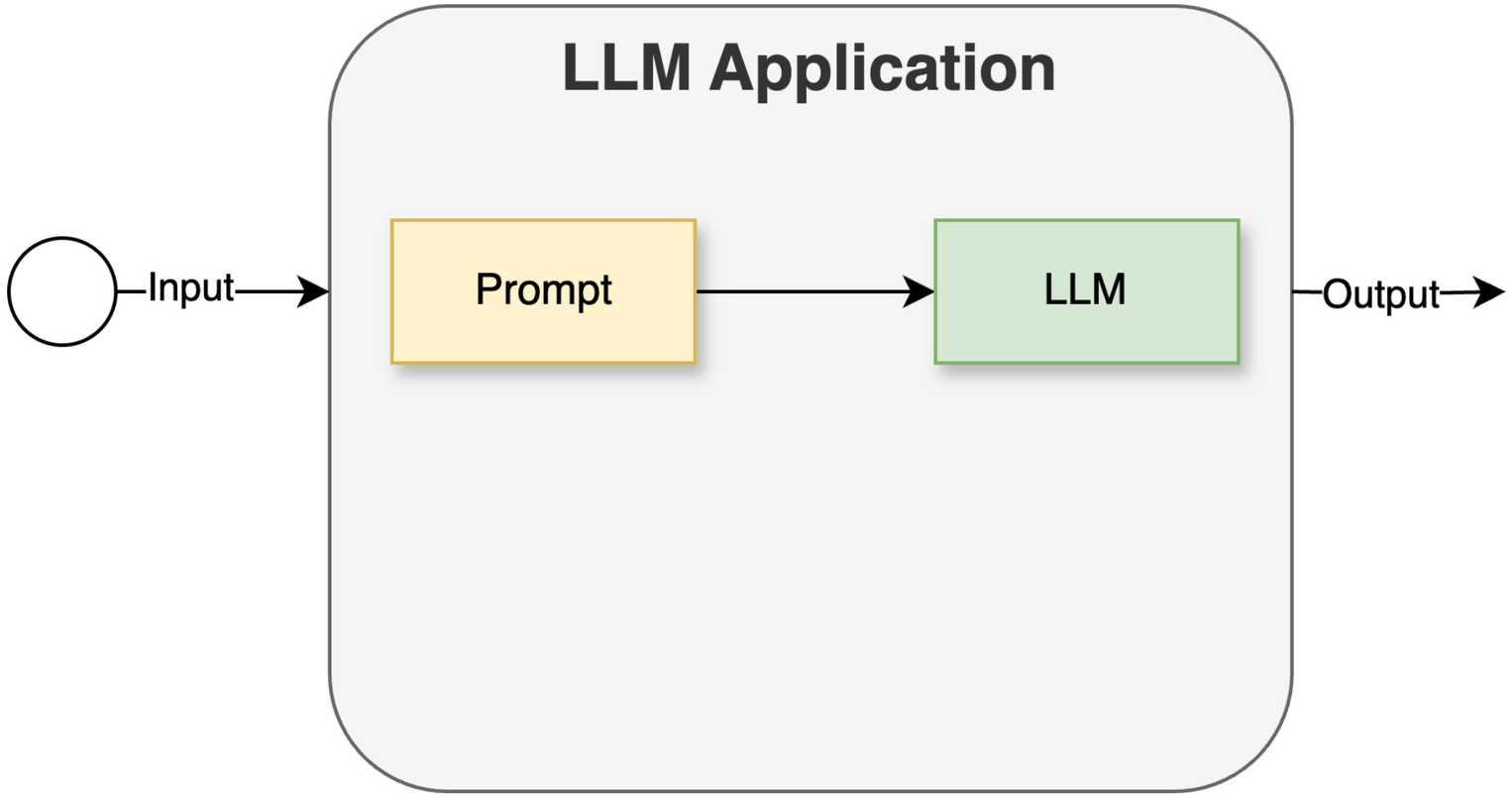
Determine the number of Cals of the dish.
Give output in "".

Examples:
{examples}

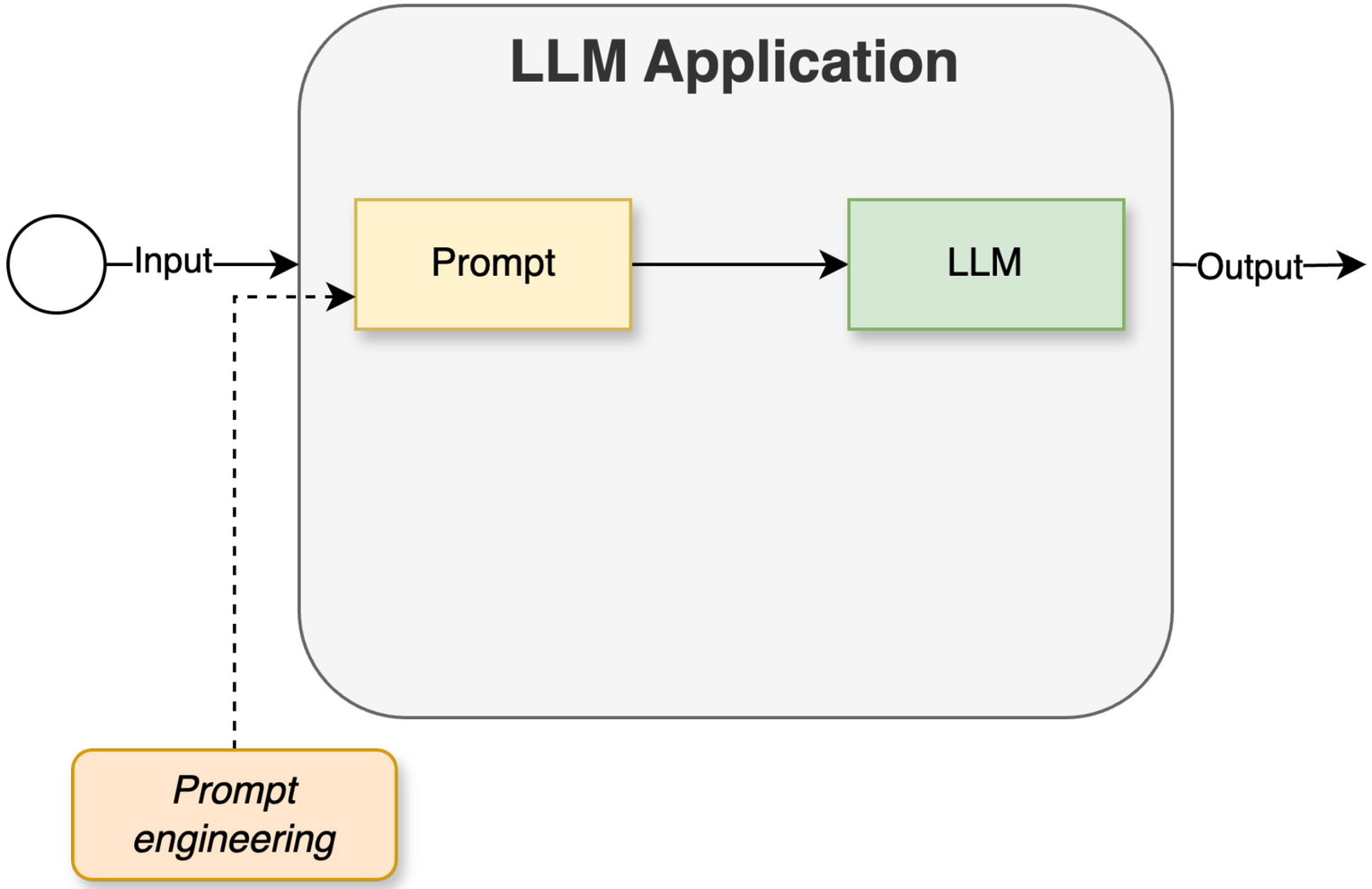
Input: {input}

Cals:

The development cycle



The development cycle



Let's practice!

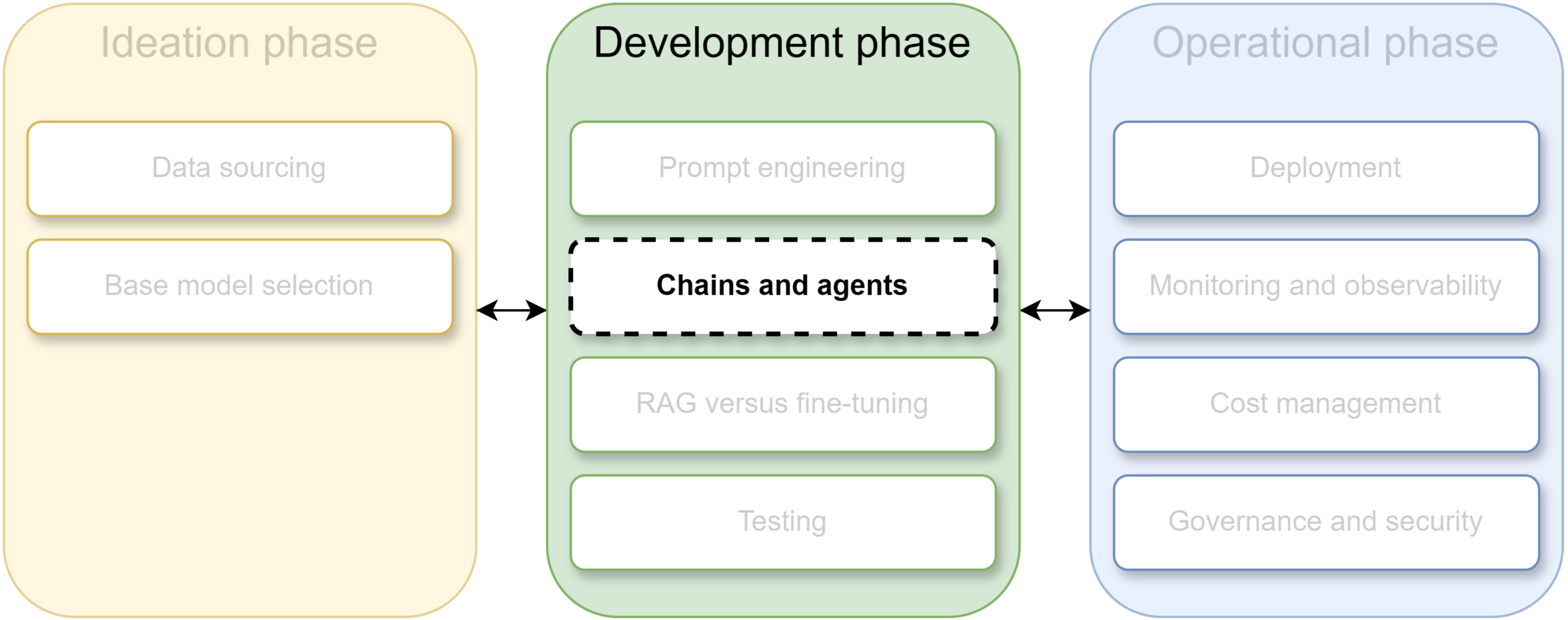
LLMOPS CONCEPTS

Chains and agents

LLMOPS CONCEPTS

Max Knobbout, PhD
Applied Scientist, Uber

LLM lifecyle: Chains and agents



From prompts to applications

Template

Determine the number of Cals of the dish.
Give output in "".

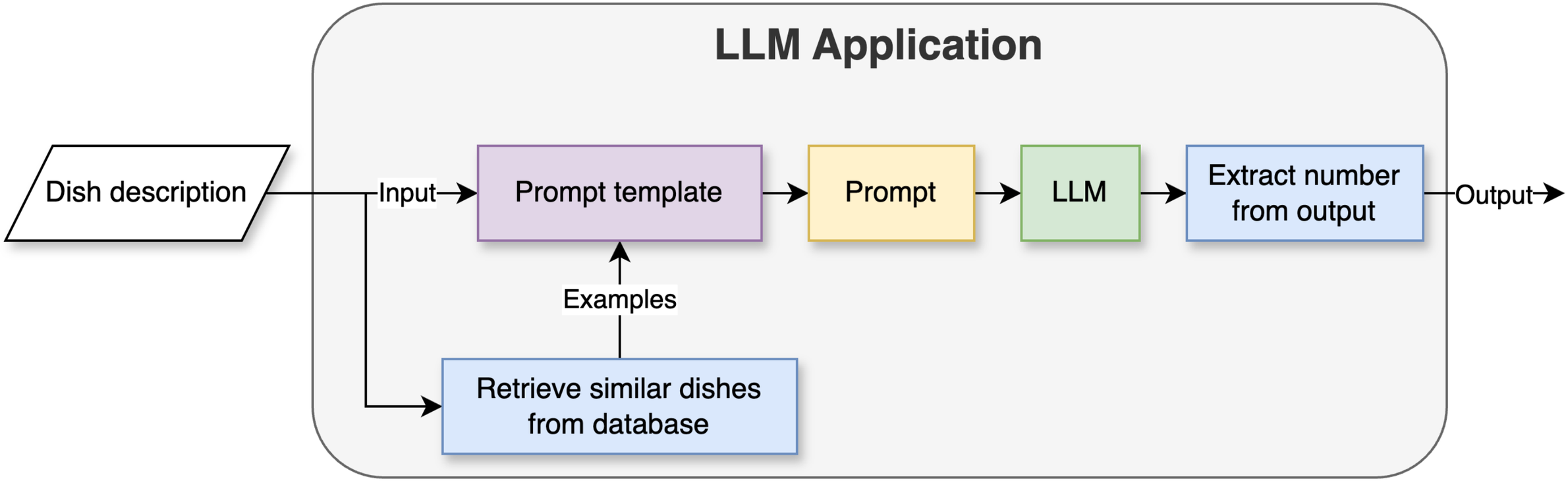
Examples:
{examples}

Input: {input}

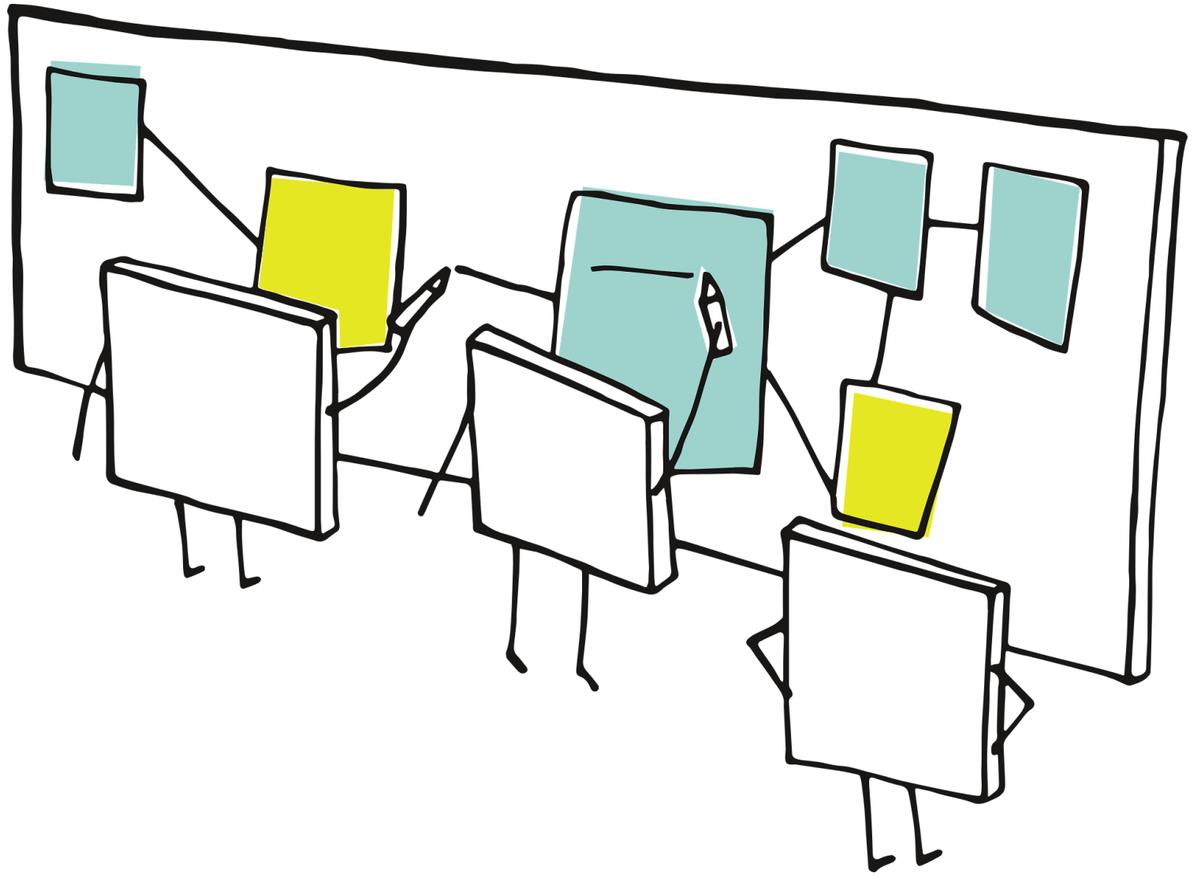
Cals:

- To use this template, we need:
  - Examples
  - Input
- We go through a few steps:
  1. Receiving input
  2. Searching examples
  3. Prompt creation
  4. Output retrieval
  5. Output parsing

A chain using our template

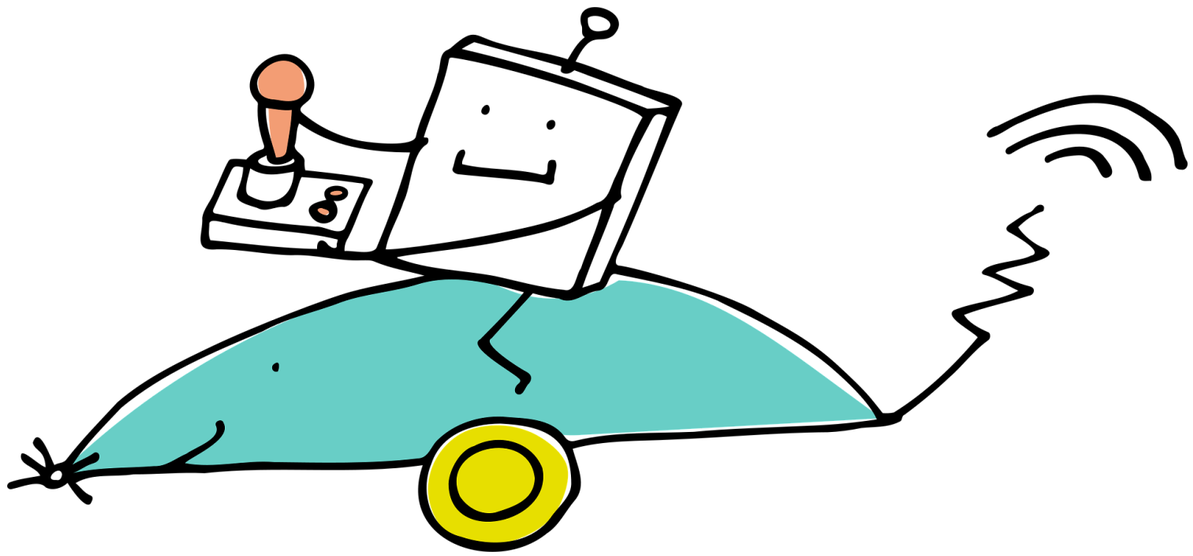


The need for chains



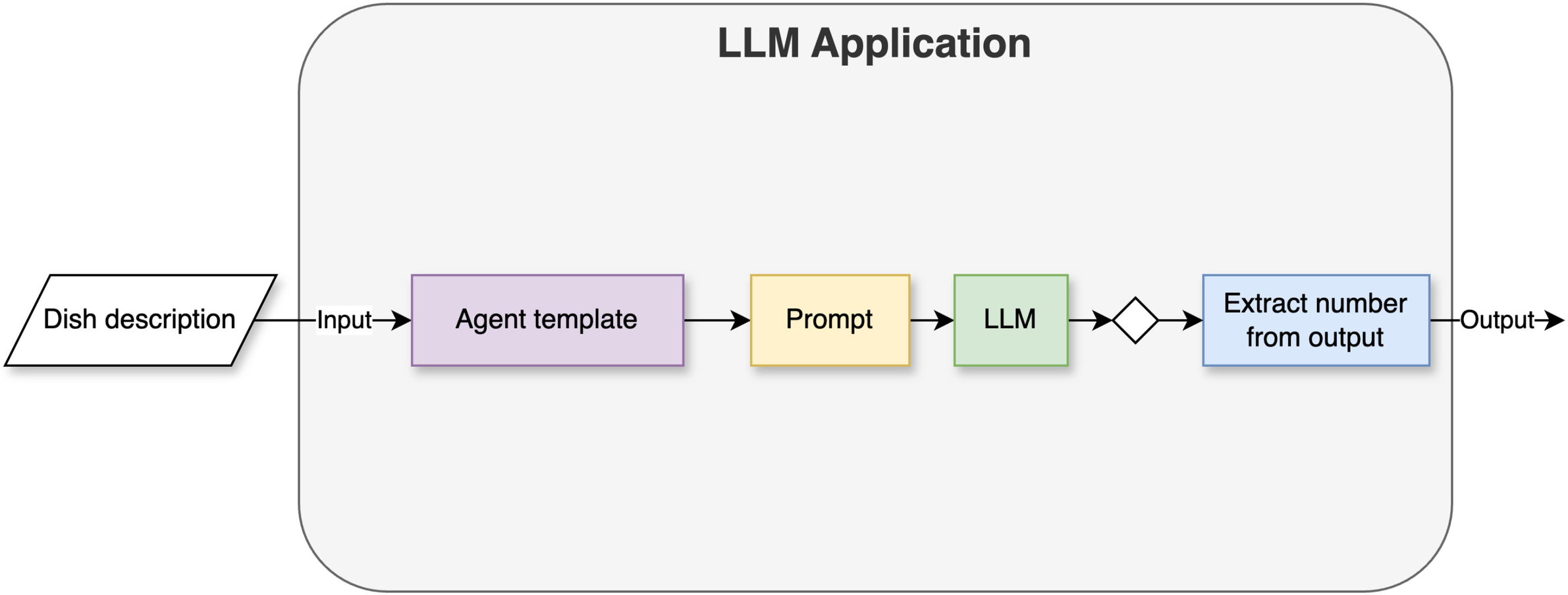
- Develop sophisticated applications
- Establish a modular design, enhancing scalability and operational efficiency
- Unlock endless possibilities for customization

Agents

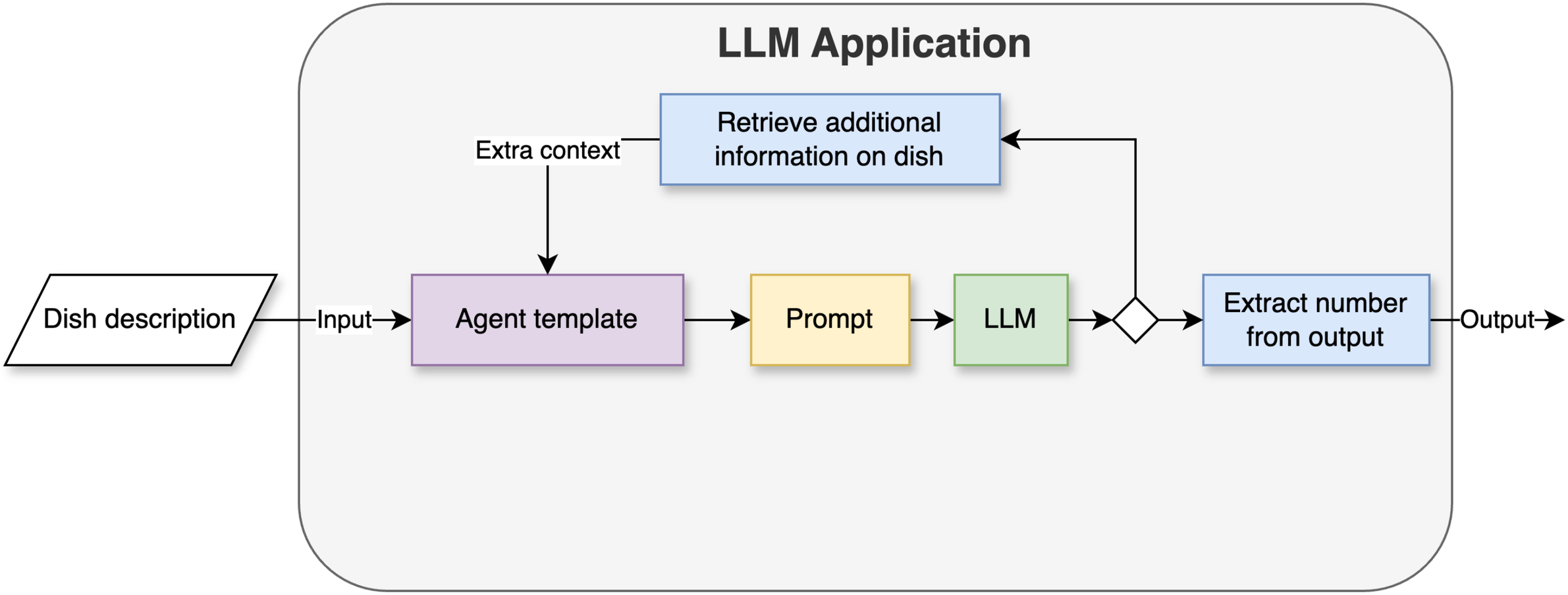


- Agents consist of:
  - Multiple actions (or tools)
  - An LLM deciding which action to take
- Useful when:
  - There are many actions
  - The optimal sequence of steps is unknown
  - We are uncertain about the inputs

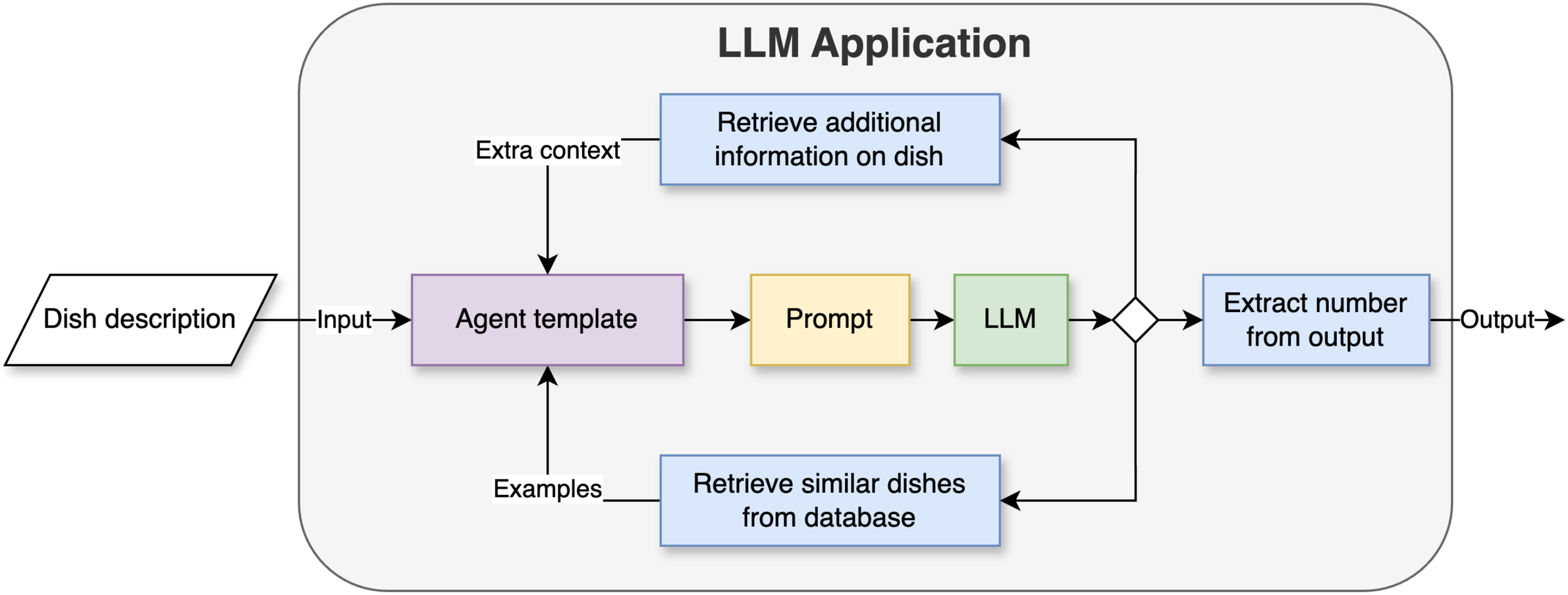
Agents


LLM Application
Dish description
Input
Agent template
Prompt
LLM
Extract number from output
Output

Agents


LLM Application
Retrieve additional information on dish
Extra context
Dish description
Input
Agent template
Prompt
LLM
Extract number from output
Output

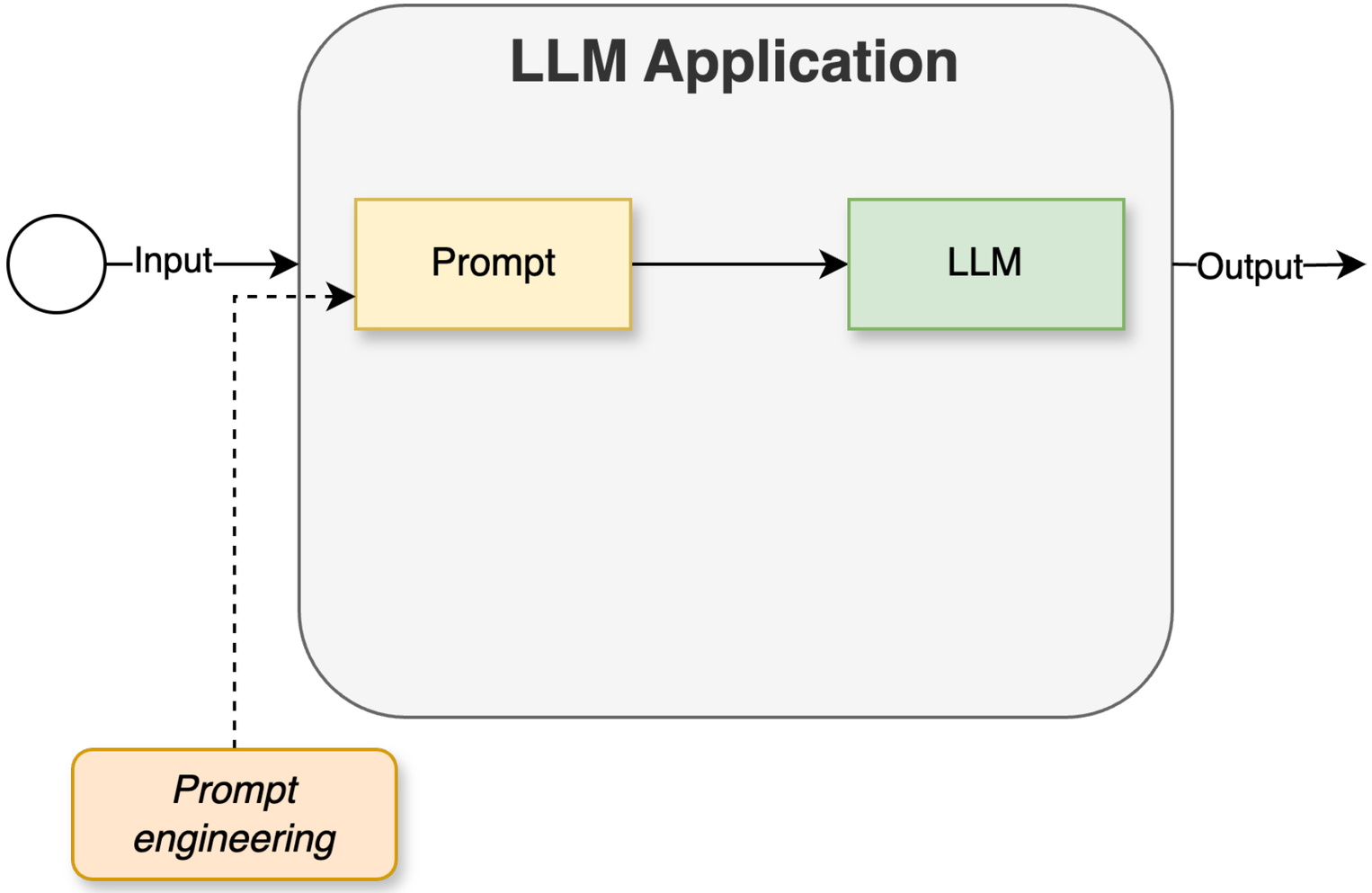
Agents


LLM Application
Retrieve additional information on dish
Extra context
Dish description
Input
Agent template
Prompt
LLM
Extract number from output
Output
Examples
Retrieve similar dishes from database

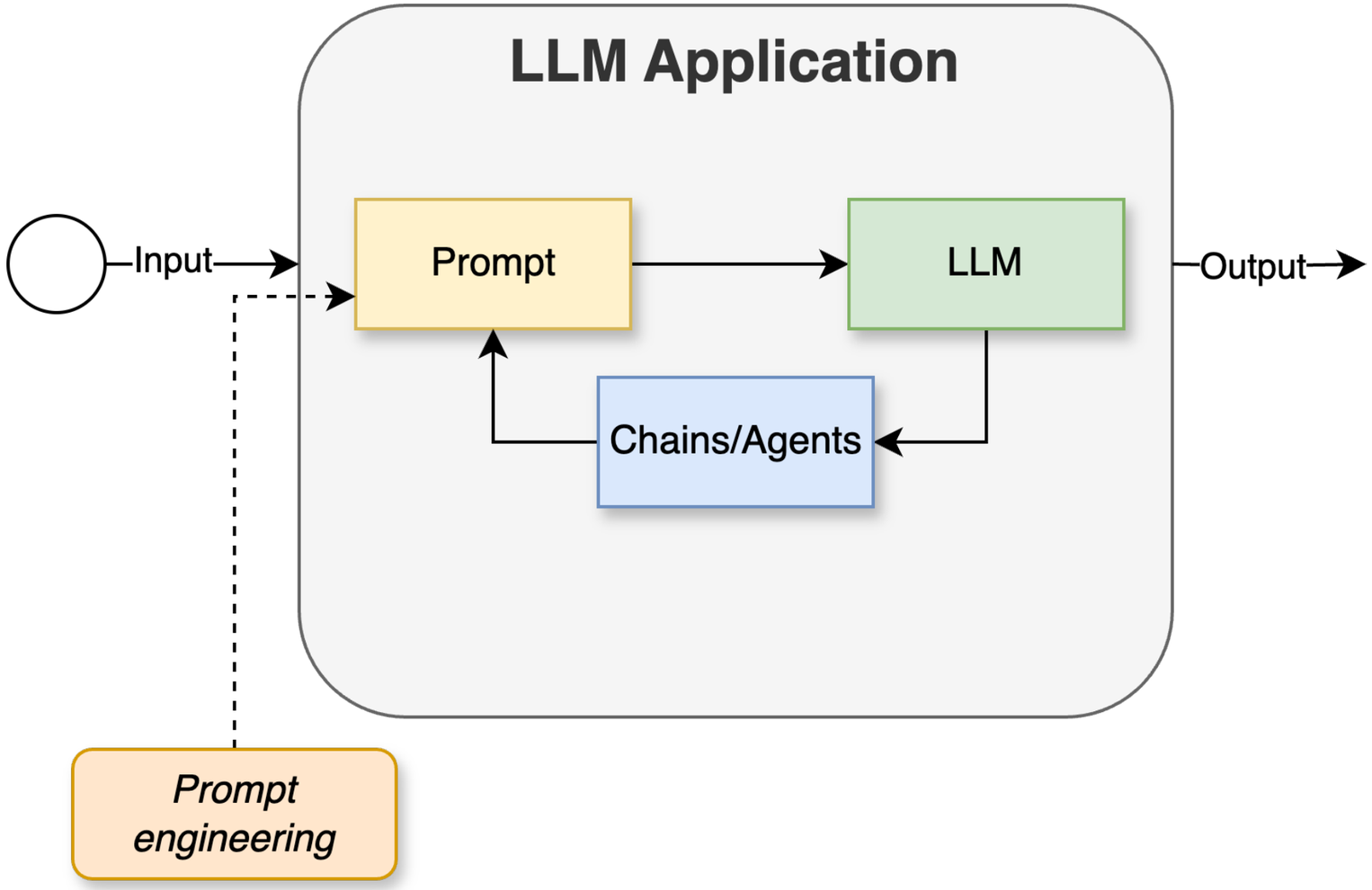
The difference between chains and agents

	Chains □	Agents □
Nature	Deterministic	Adaptive
Complexity	Low	High
Flexibility	Low	High
Risk	Lower (due to predictability)	Higher (due to adaptability)

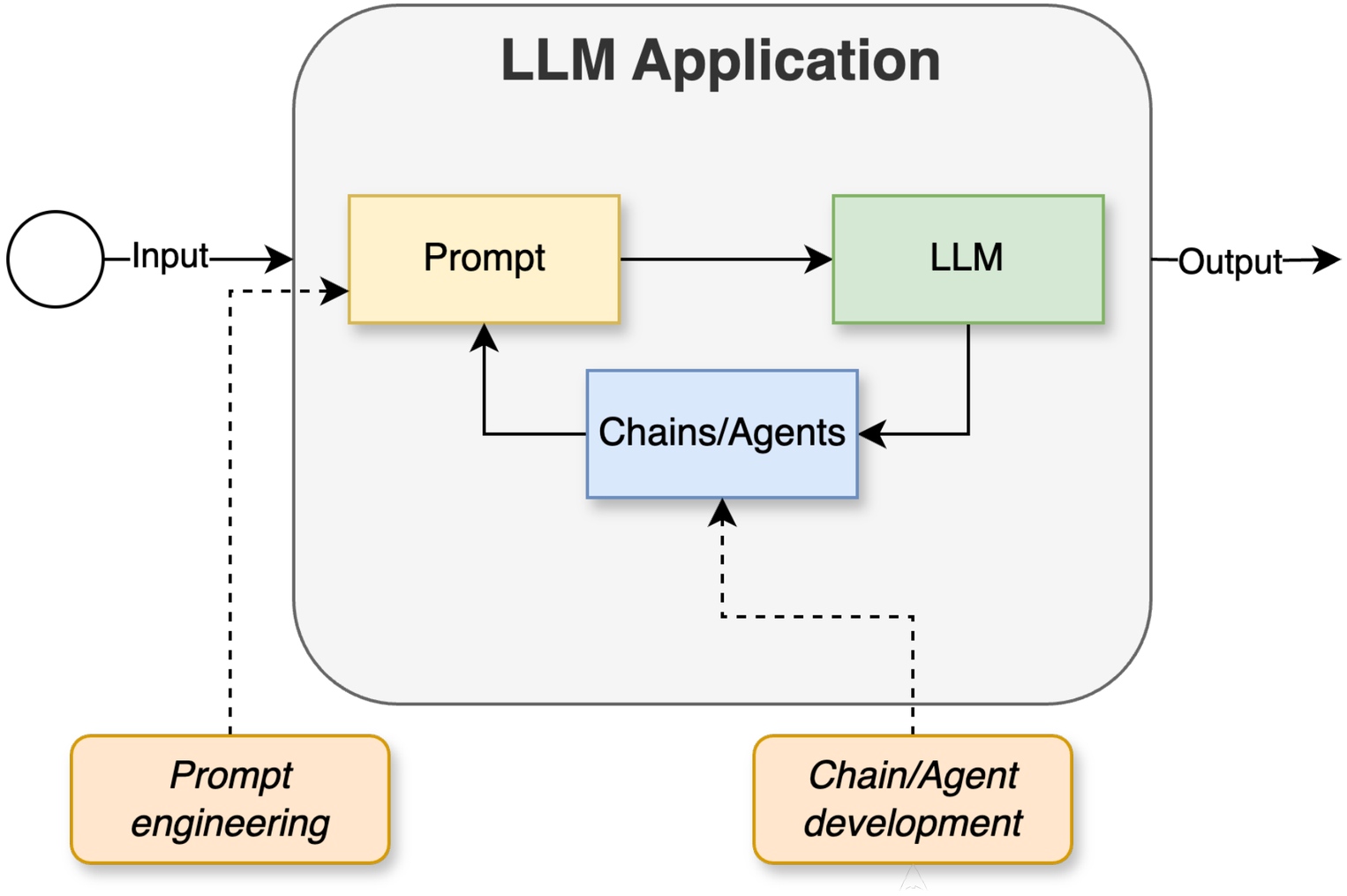
The development cycle



The development cycle



The development cycle



Let's practice!

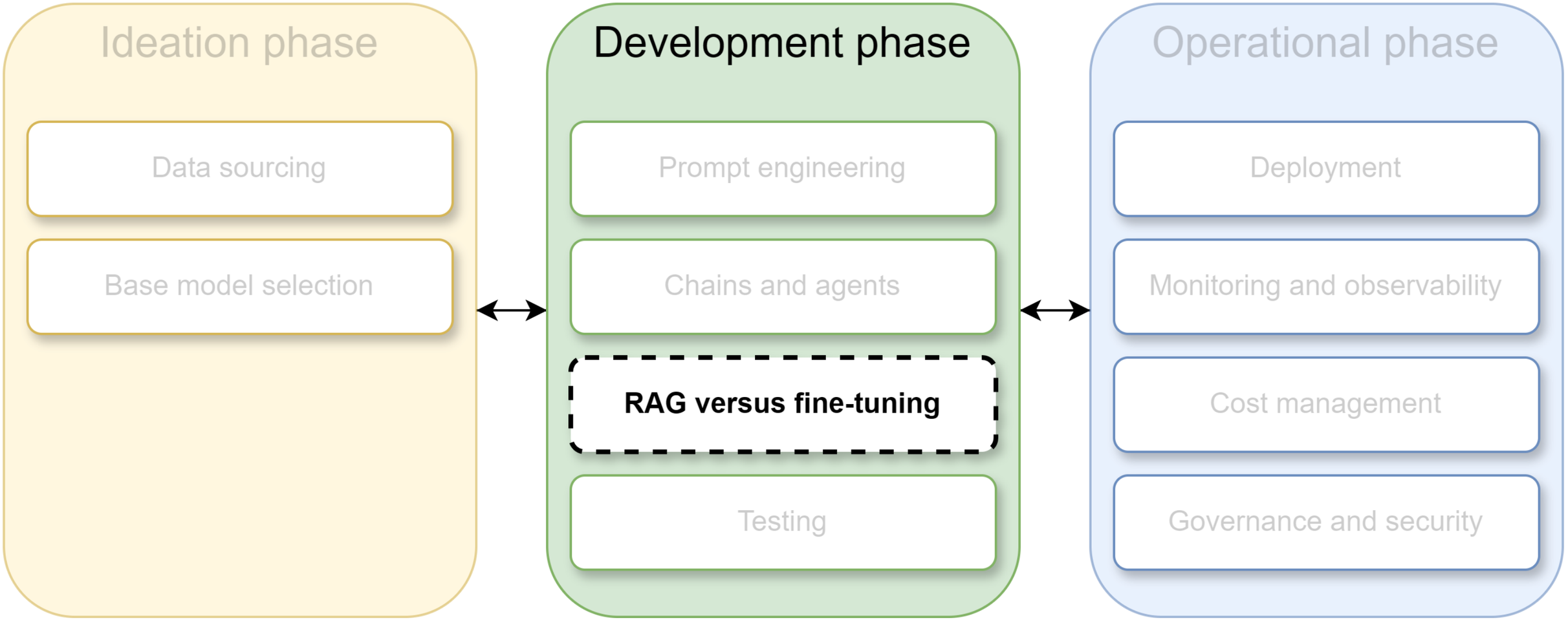
LLMOPS CONCEPTS

RAG versus fine-tuning

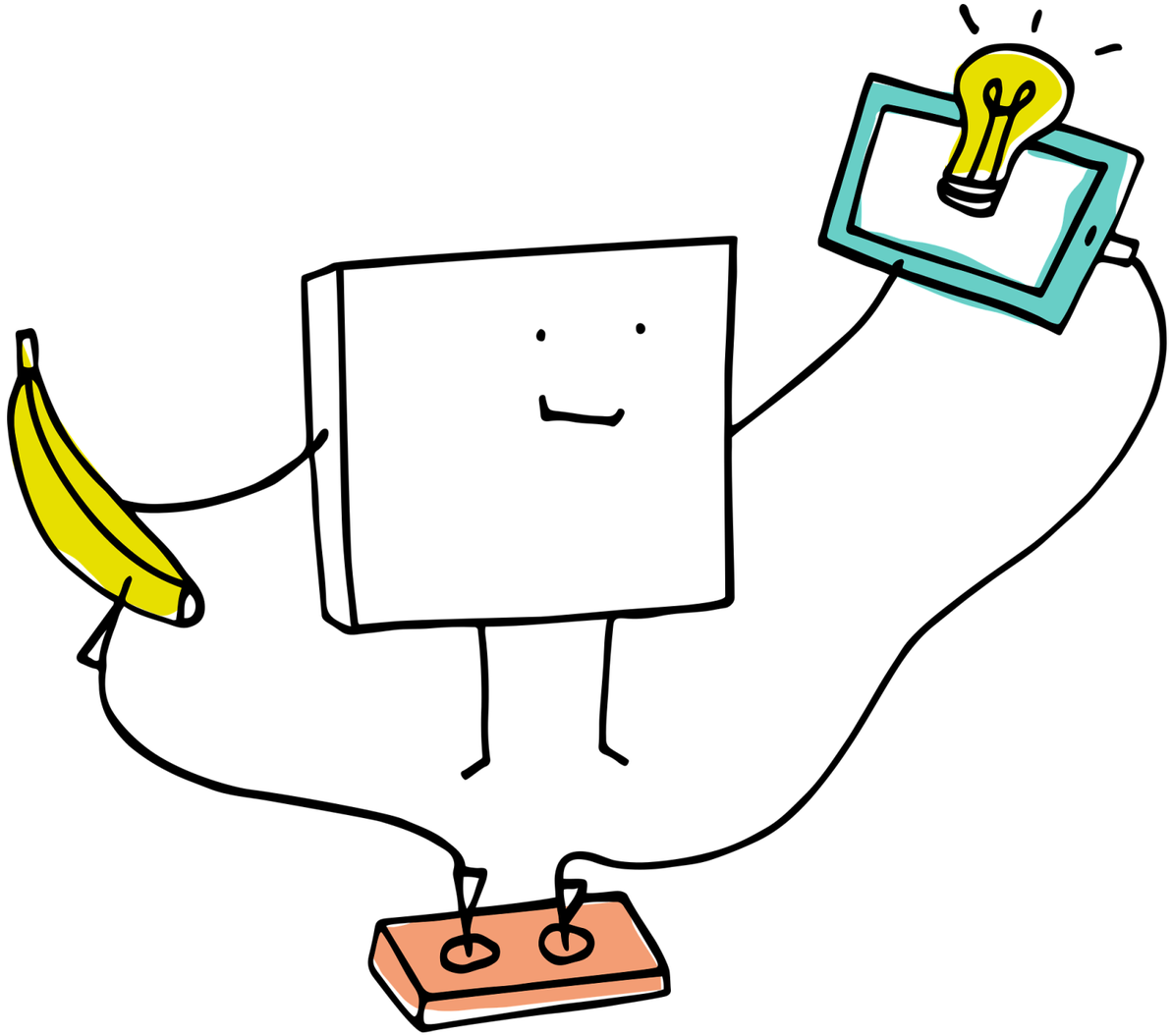
LLMOPS CONCEPTS

Max Knobbout, PhD
Applied Scientist, Uber

LLM lifecyle: RAG versus fine-tuning



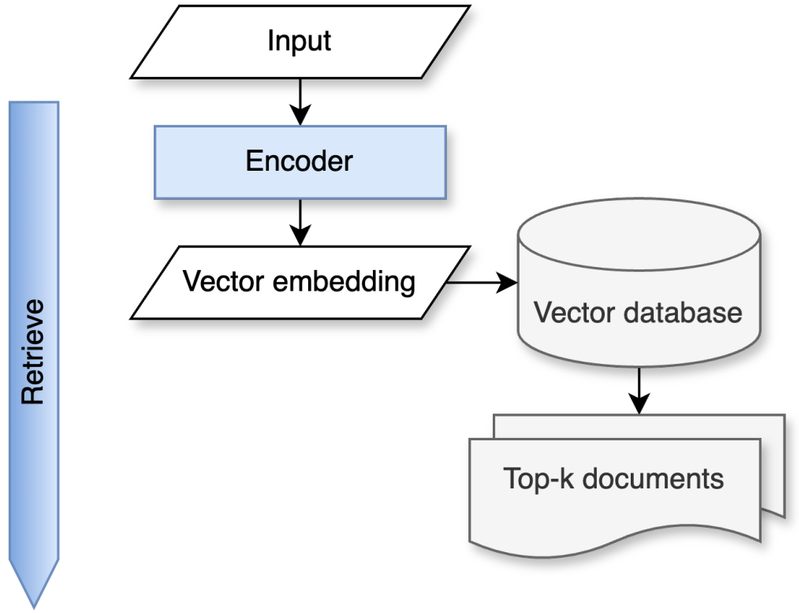
Retrieval Augmented Generation (RAG)



- Combine LLMs' reasoning capabilities with external knowledge.
- Three steps in a chain:
  1. **Retrieve** related documents
  2. **Augment** prompt with examples
  3. **Generate** output
- Often implemented using vector databases.

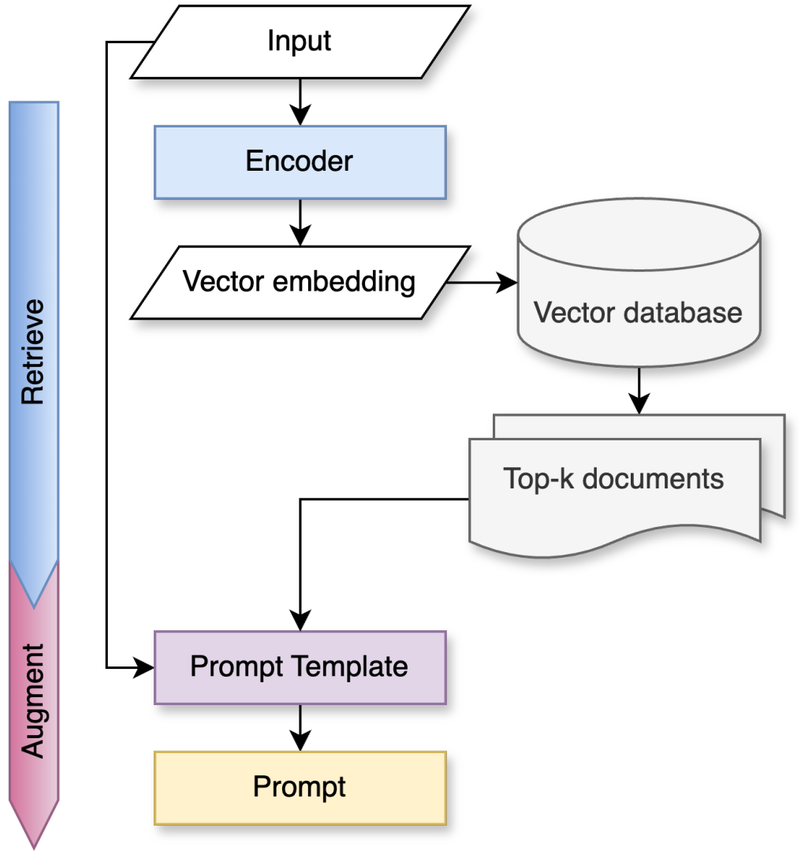
RAG-chain with vector database

1. **Retrieve:**
   - Convert input to embedding
   - Search vector database
   - Retrieve most similar documents



RAG-chain with vector database

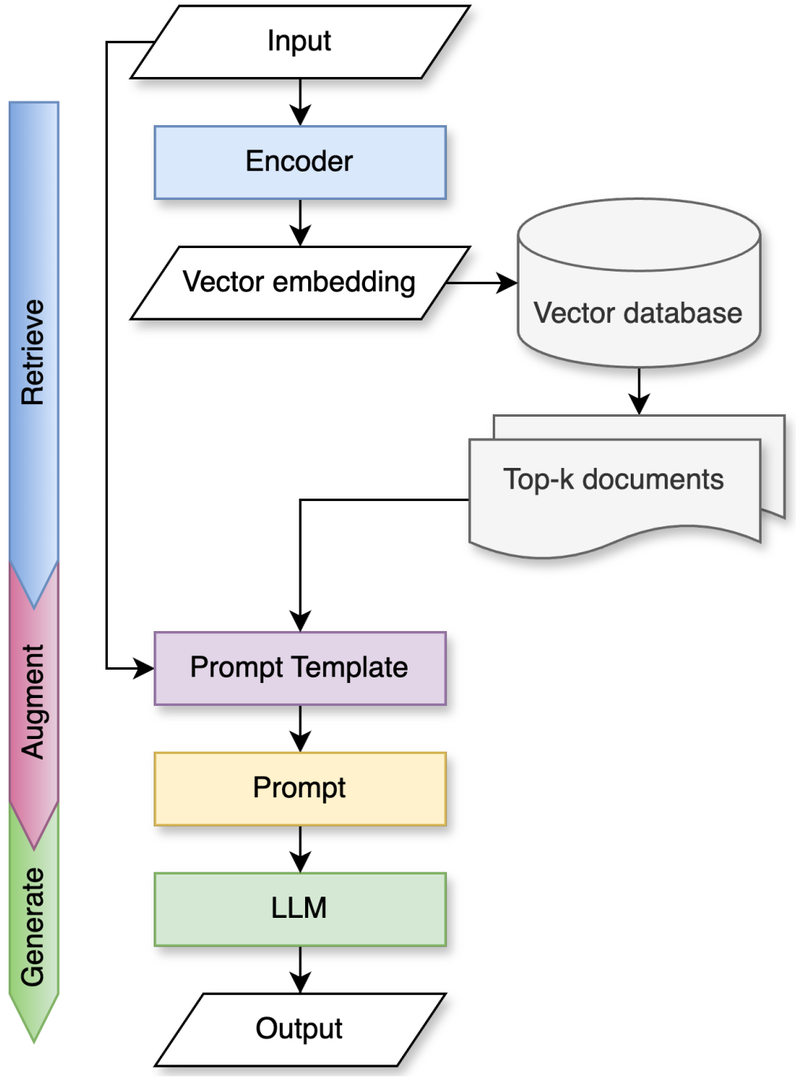
1. **Retrieve:**
   - Generate embedding from input
   - Search vector database
   - Retrieve most similar documents
2. **Augment:**
   - Combines input with documents to create final prompt



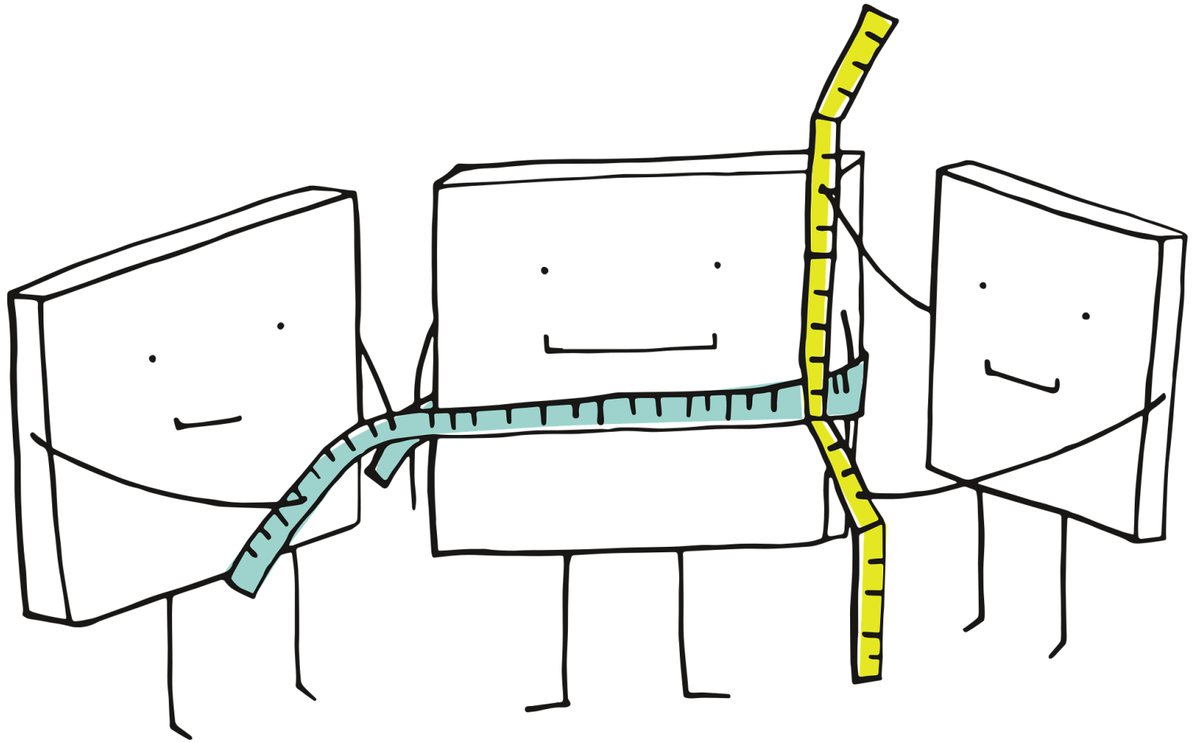
RAG-chain with vector database

1. **Retrieve:**
   - Generate embedding from input
   - Search vector database
   - Retrieve most similar documents
2. **Augment:**
   - Combine input with top-k documents and create augmented prompt
3. **Generate:**
   - Uses prompt to create an output

Many implementation choices and embedding models. **Experiment and test!**



Fine-tuning



- Adjusts the LLM's weights
- Expand to specific tasks and domains:
  - Different languages
  - Specialized fields

Fine-tuning

Supervised fine-tuning (transfer learning)

Type of data needed :

- Demonstration data (inputs with desired outputs)

Approach :

- Re-train (parts of) the model

Reinforcement Learning from Human Feedback (RLHF)

Type of data needed :

- Rankings or quality scores (obtained from likes & dislikes)

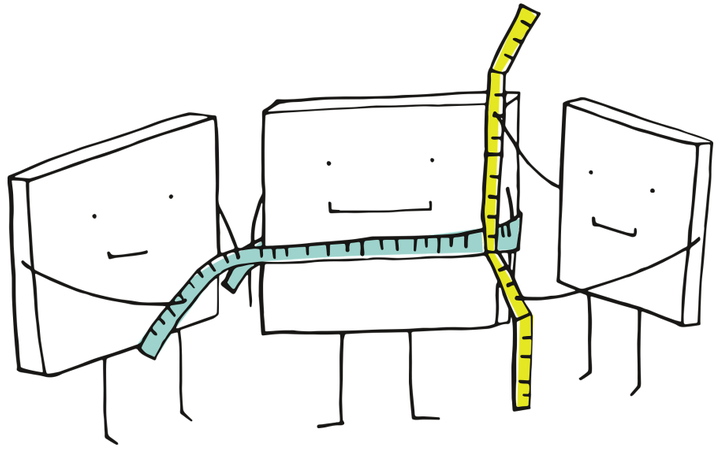
Approach :

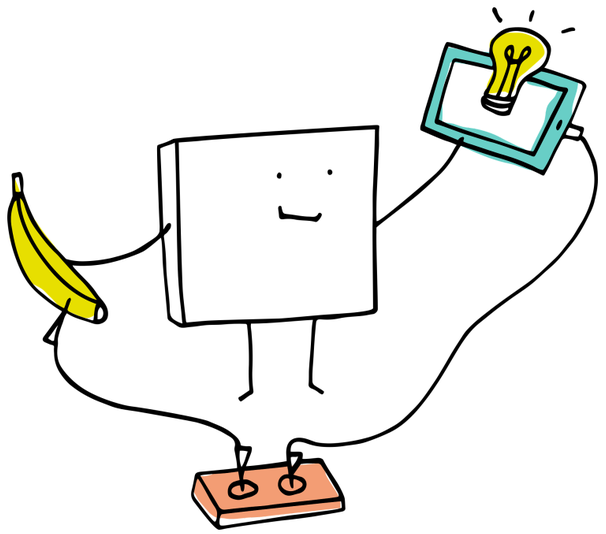
- Train an extra reward model
- Optimize original LLM to maximize this

RAG or fine-tuning

RAG

- Use when including factual knowledge
- Keeps capabilities of LLM, easy to implement, always up-to-date
- Adds extra components, requires careful engineering

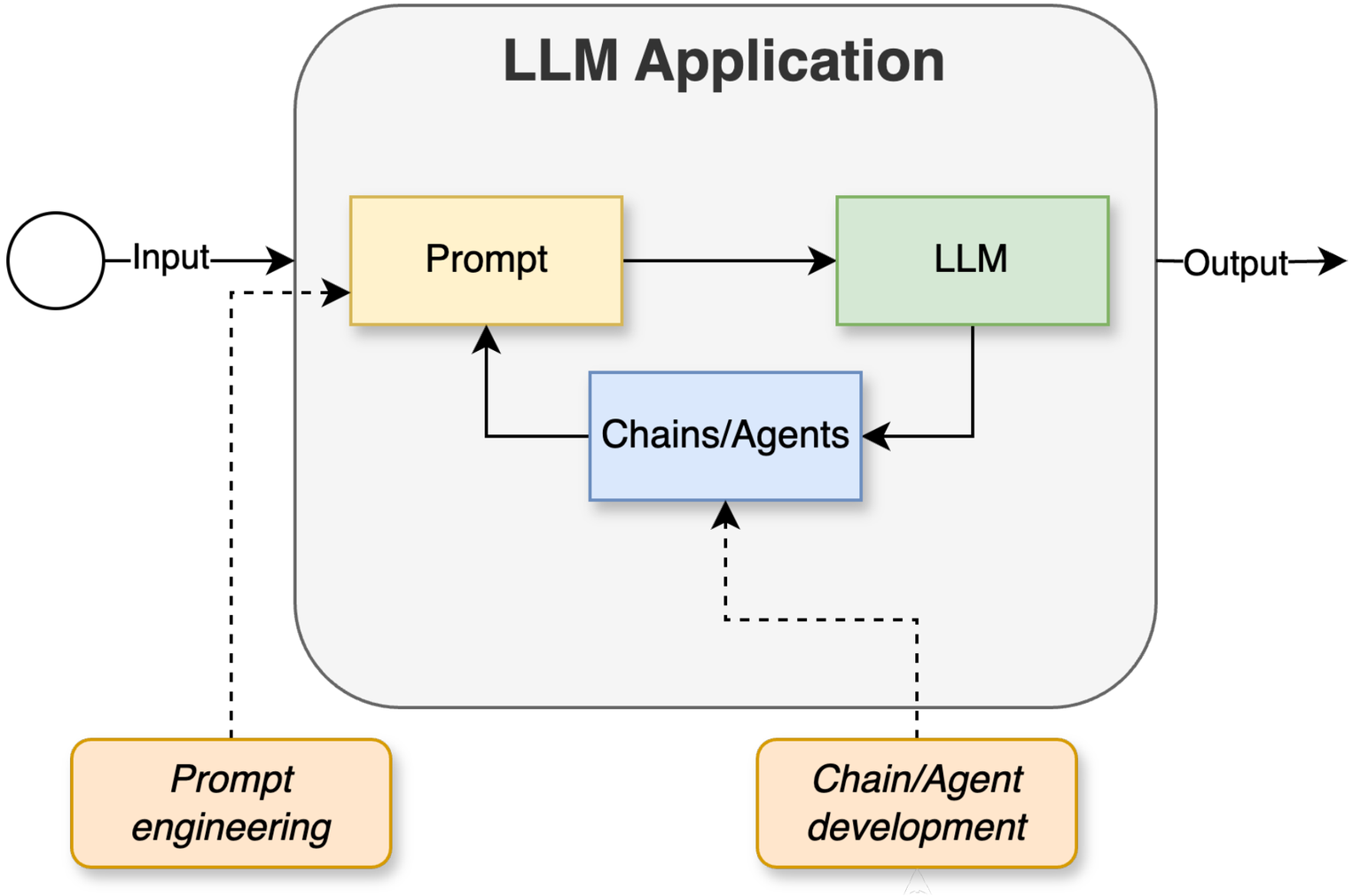




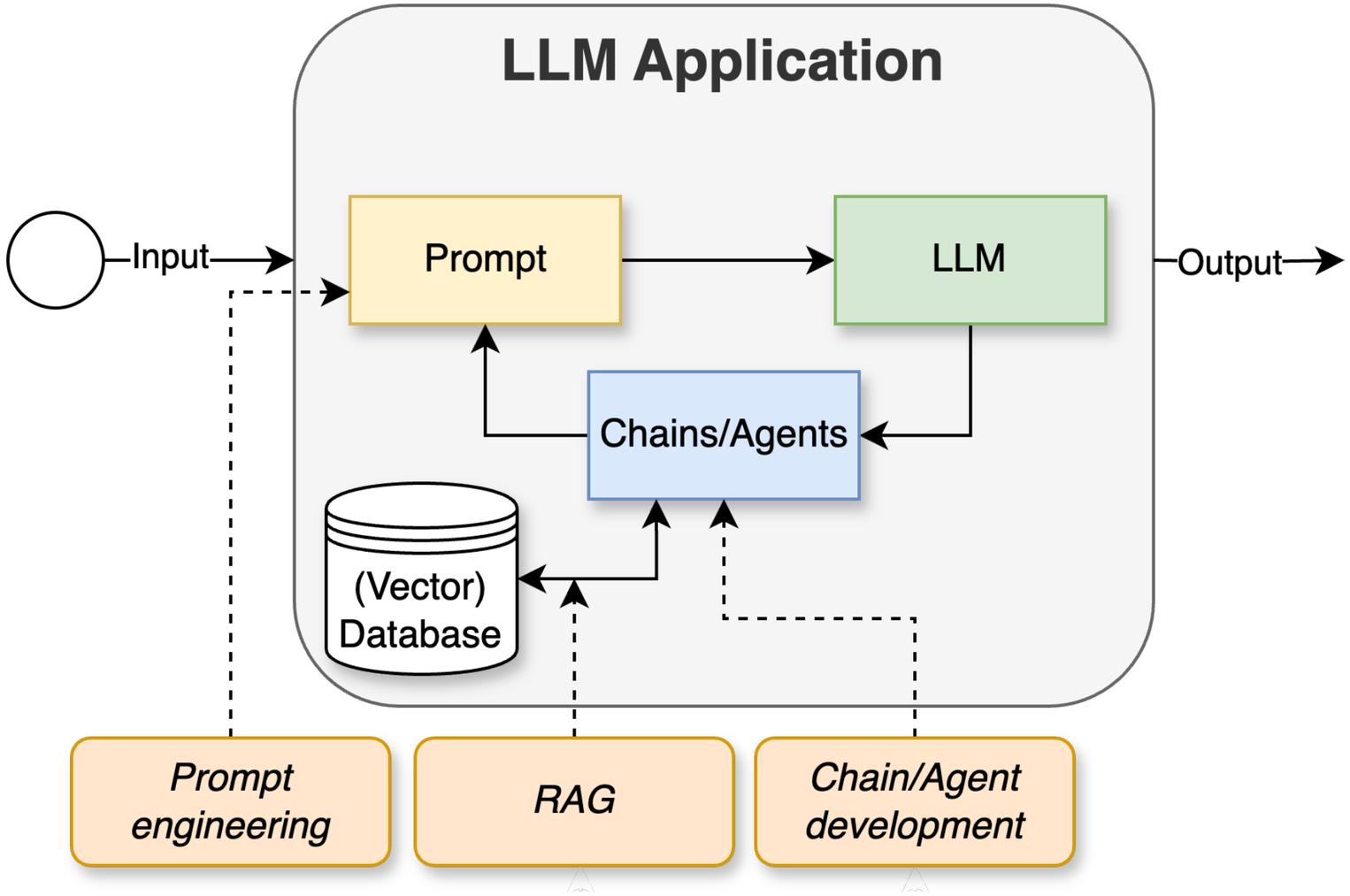
Fine-tuning

- Use when specializing in new domain
- Full control and no extra components
- Needs labeled data & specialized knowledge, bias amplification, catastrophic forgetting

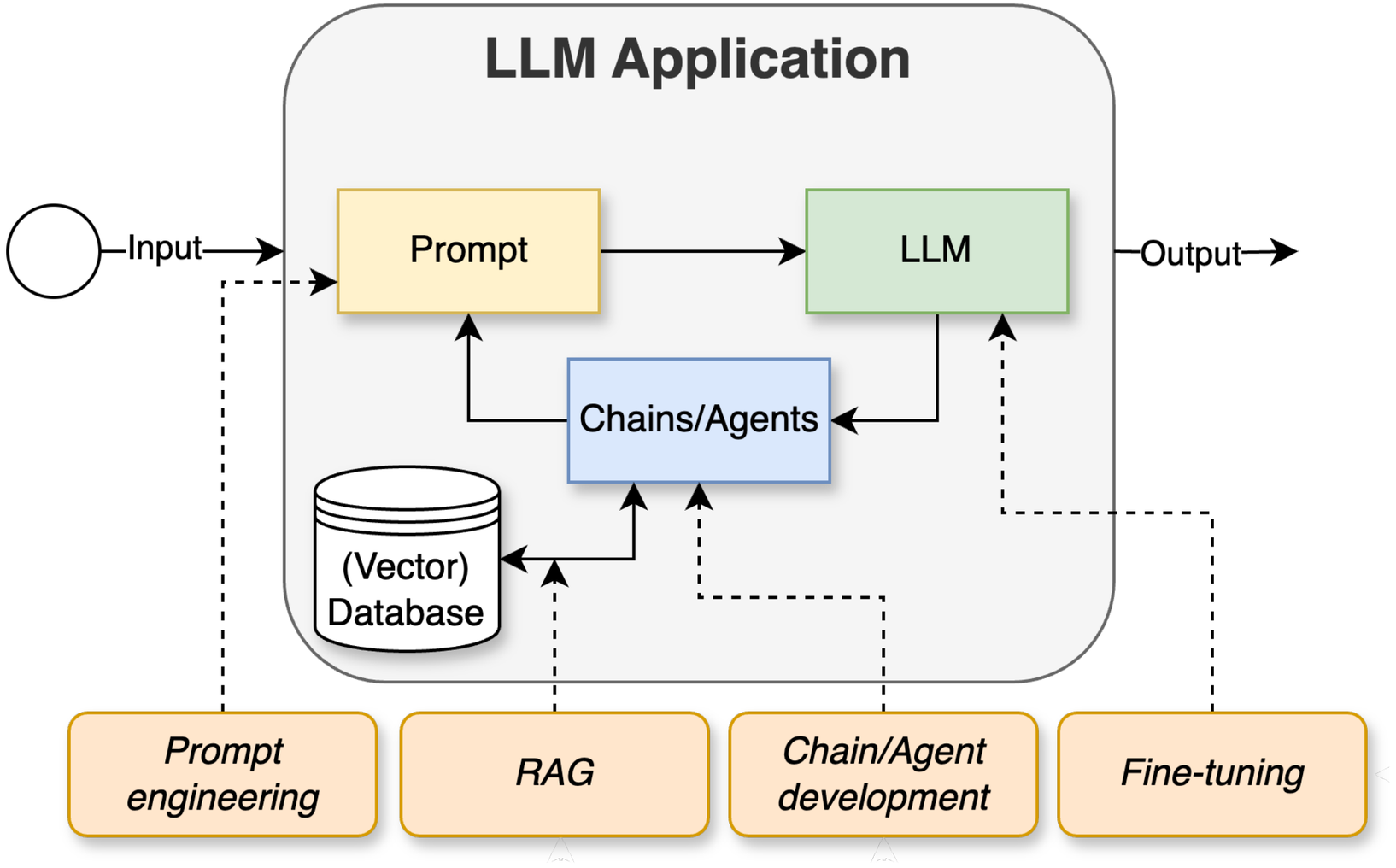
The development cycle



The development cycle



The development cycle



Let's practice!

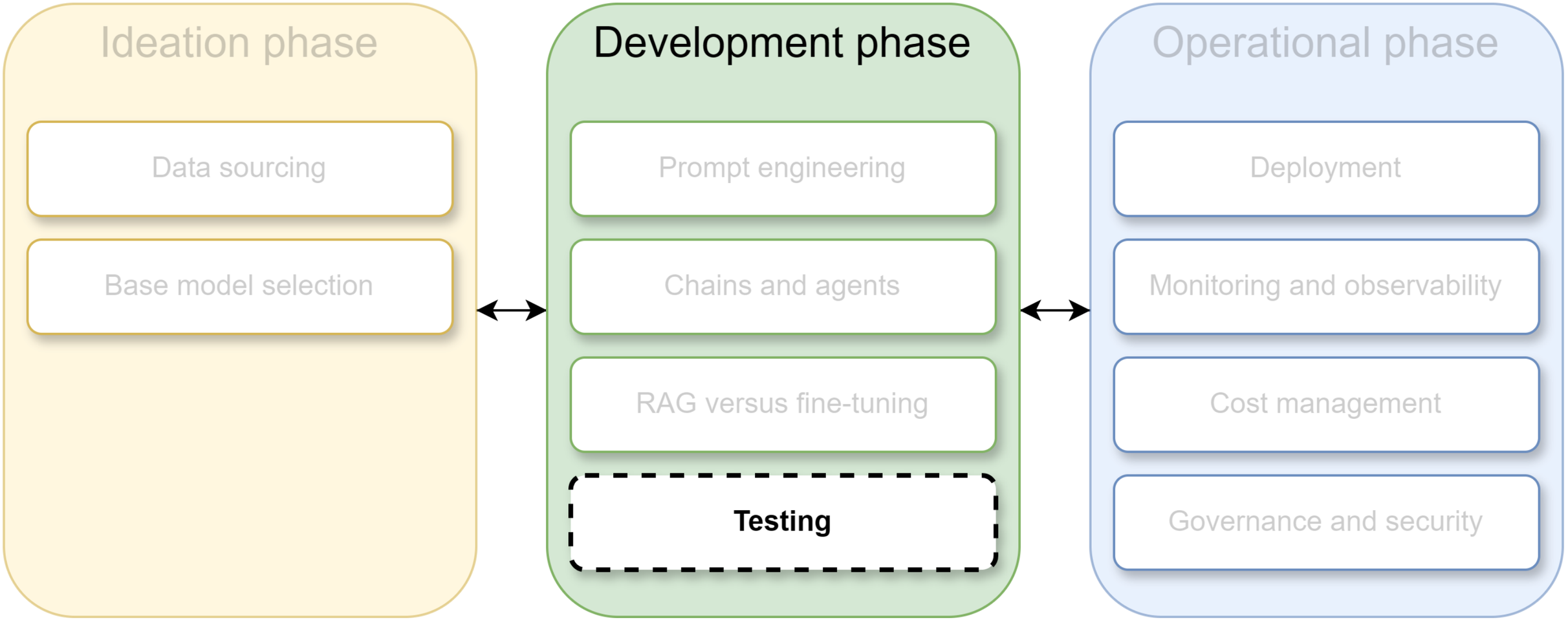
LLMOPS CONCEPTS

Testing

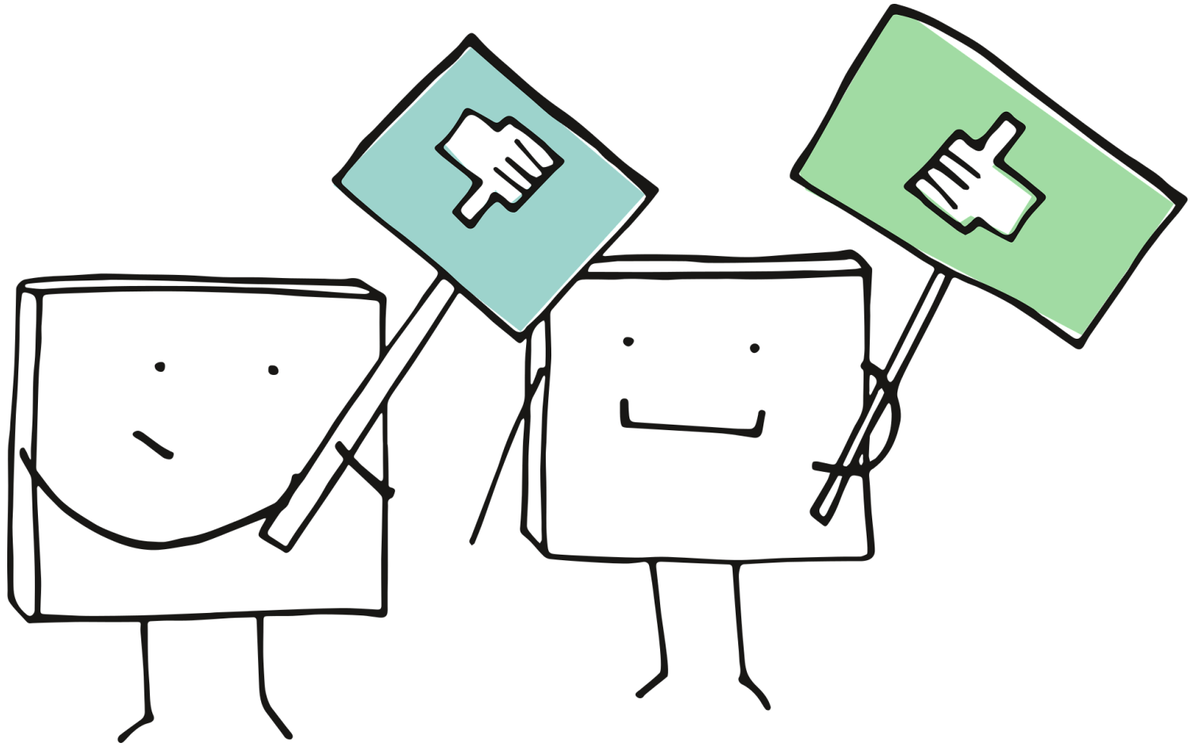
LLMOPS CONCEPTS

Max Knobbout, PhD
Applied Scientist, Uber

LLM lifecyle: Testing



Why do we need to test?

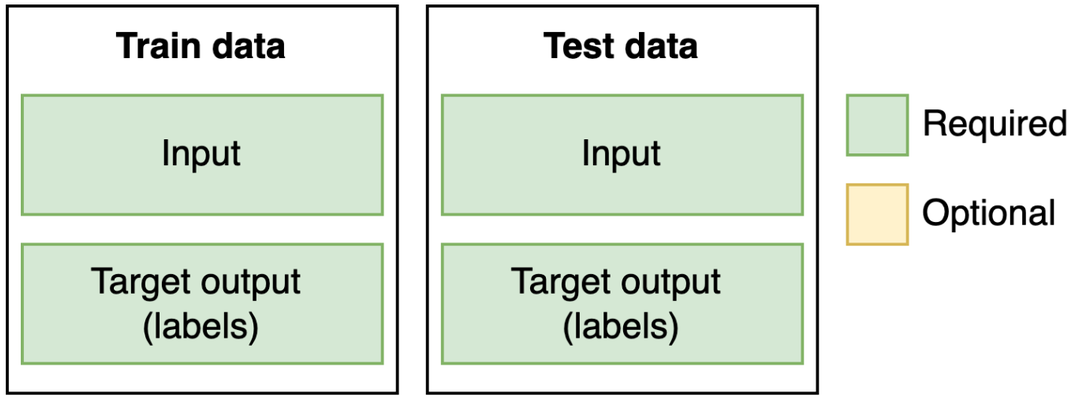


- LLMs make mistakes
- Testing is vital for assessing the application's readiness for deployment
- We will address evaluating the **output**

Traditional ML versus LLM application testing

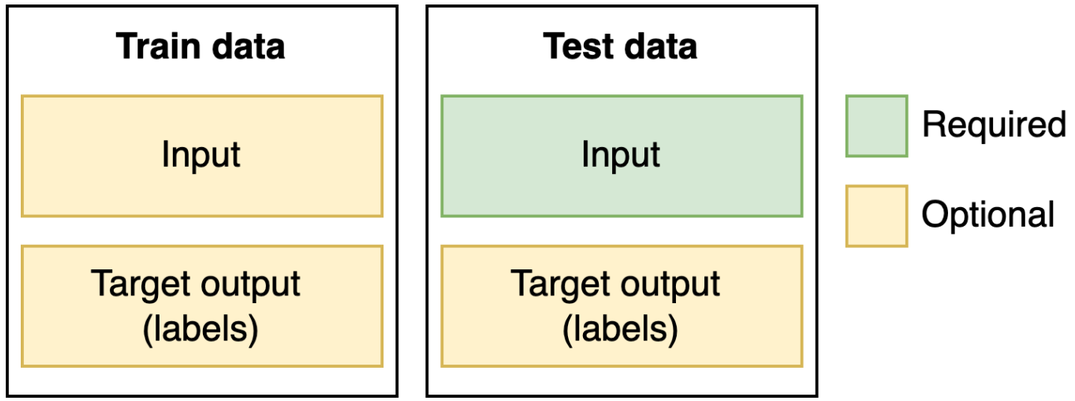
Traditional supervised machine learning:

- Need labeled train and test data
- Metrics focusing on accuracy or closeness to target

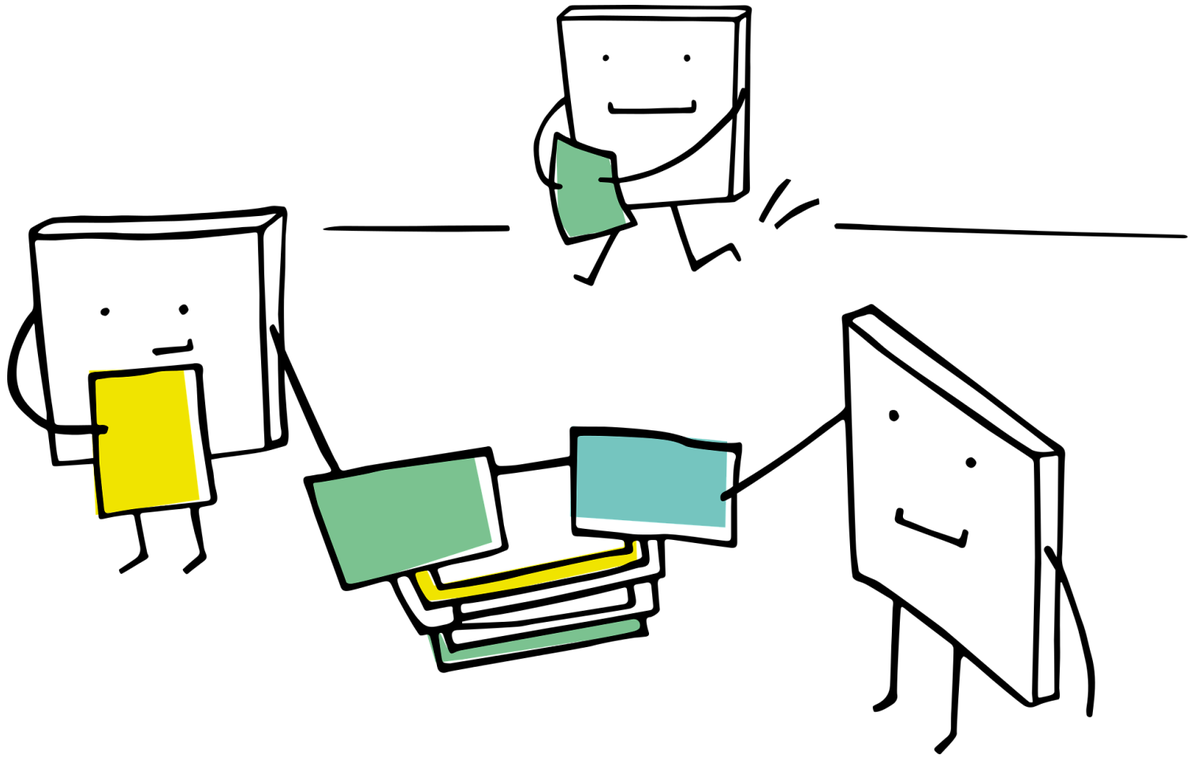


LLM applications:

- Need test data, not necessarily labeled
- Quality of output using a variety of metrics



Step 1: Building a test set

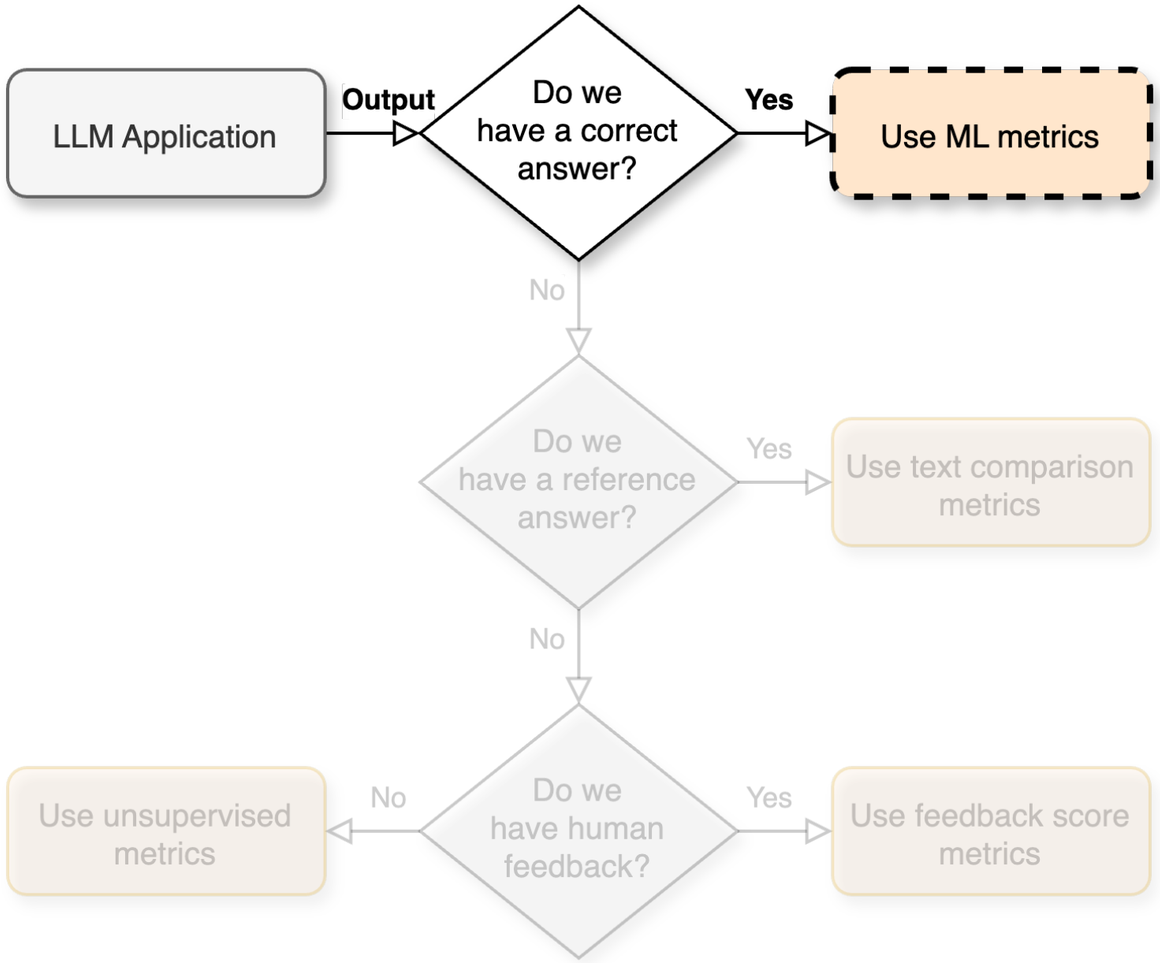


- Building the test set should now be completed
- Test data must closely resemble real-world scenarios
- Various tools can help us in this process

Step 2: Choosing our metric

If there is a correct answer...

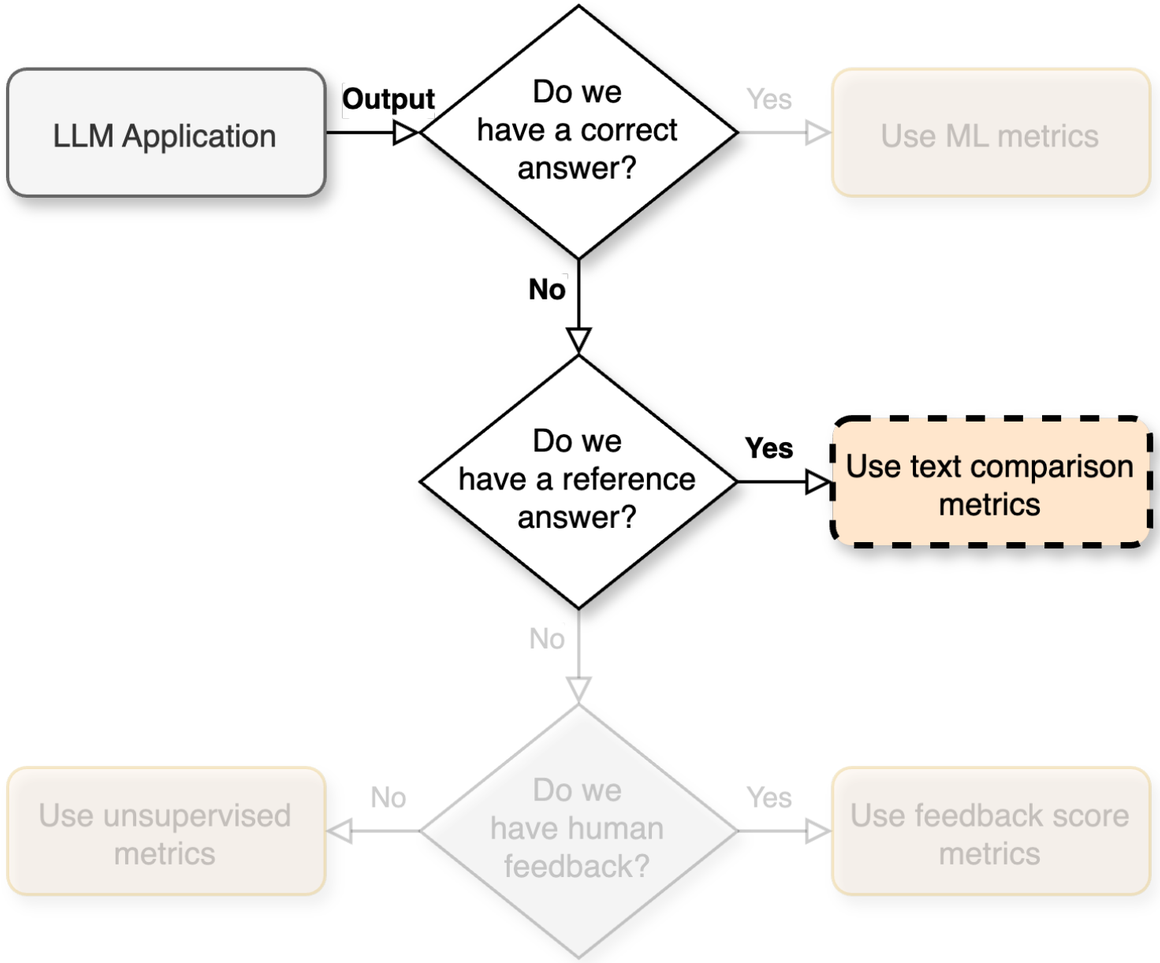
- ... use machine learning metrics. Example:
  - Accuracy



Step 2: Choosing our metric

If there is a reference answer...

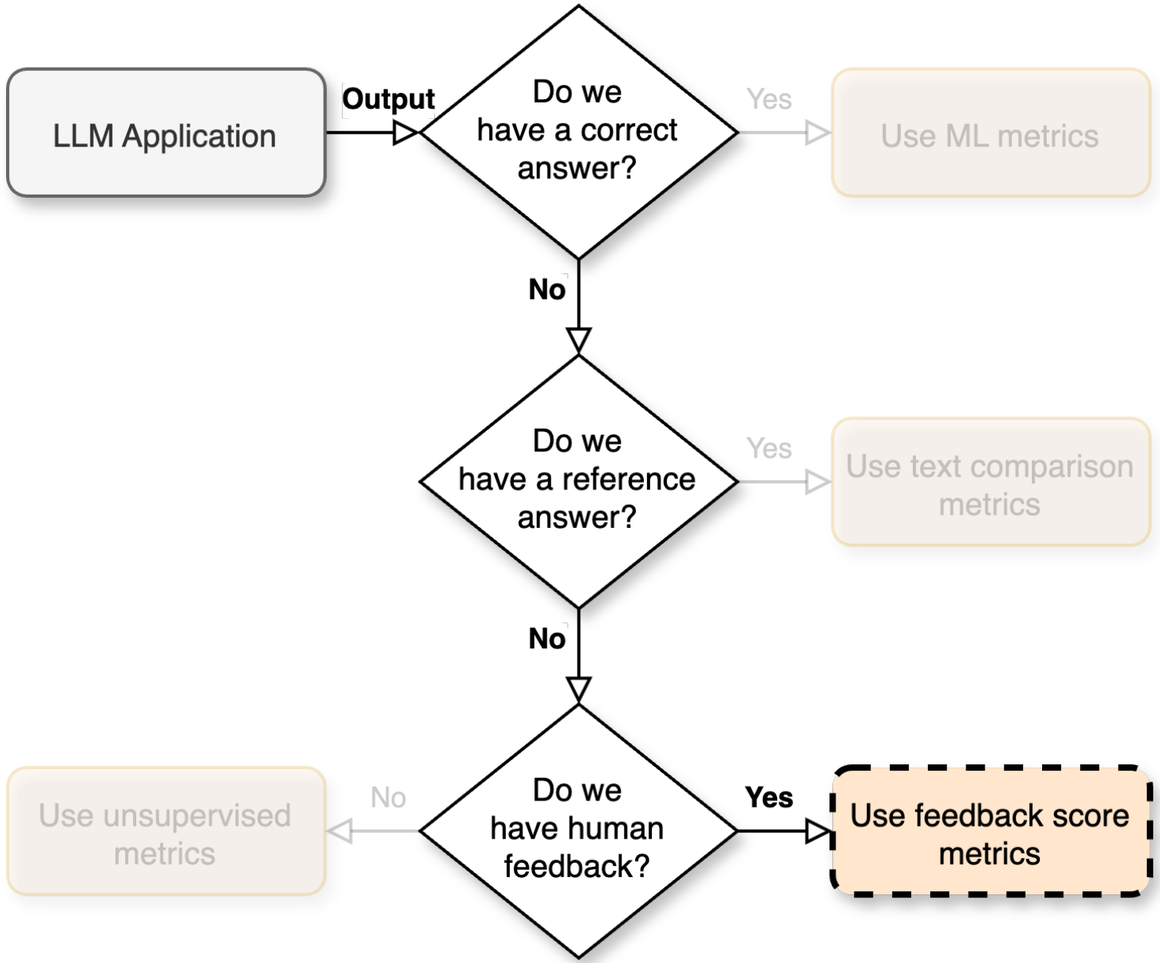
- ... use statistical methods.
- ... use model-based methods. Example:
  - LLM judges



Step 2: Choosing our metric

If we have access to human feedback...

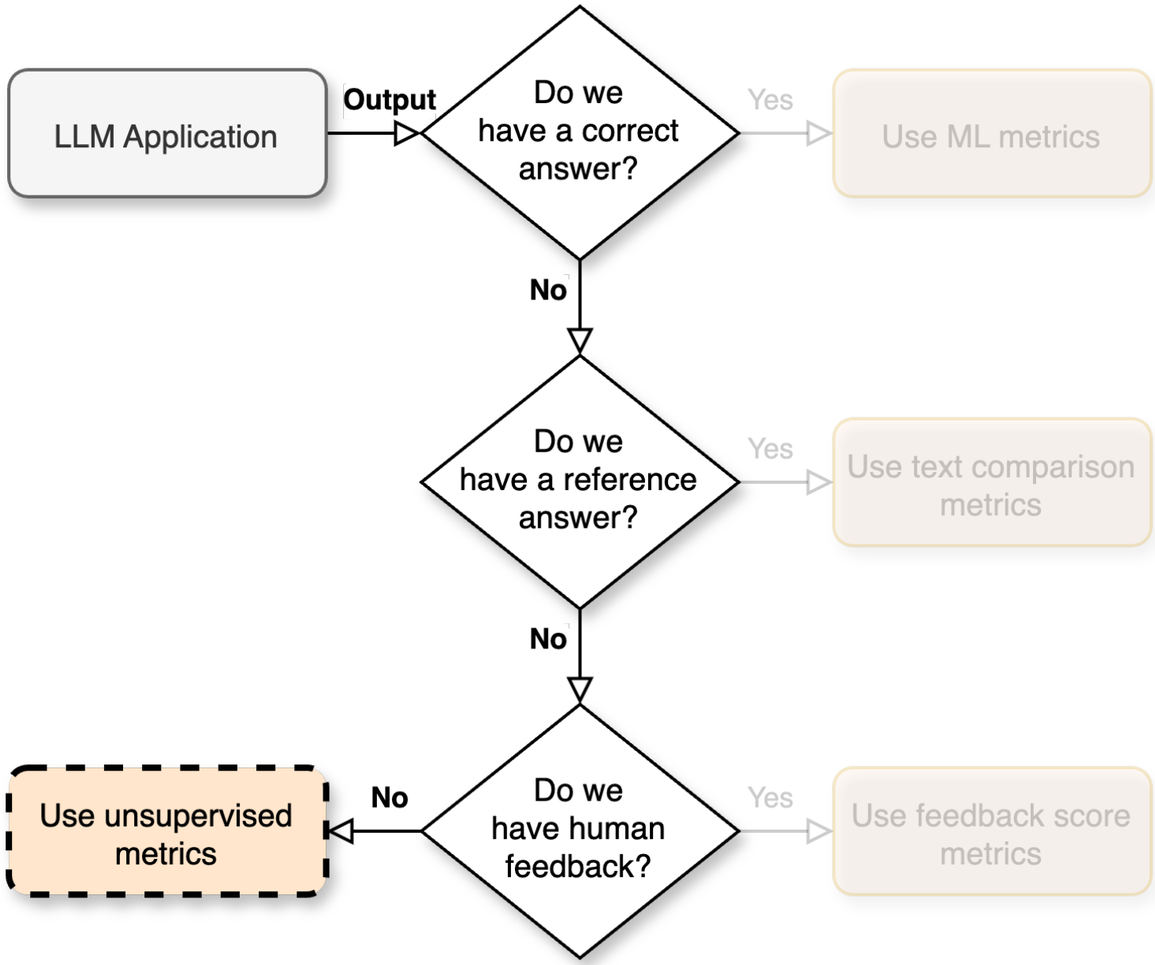
- ... let humans rate the text. Examples:
  - Rate quality
  - Rate relevance
  - Rate coherence
- ... use model-based approach. Example:
  - Predict rating based on past feedback
  - Ask LLM judge if feedback was incorporated



Step 2: Choosing our metric

If there's no human feedback...

- ... use unsupervised metrics. Examples:
  - Coherence
  - Fluency
  - Diversity



Step 3: Define optional secondary metrics

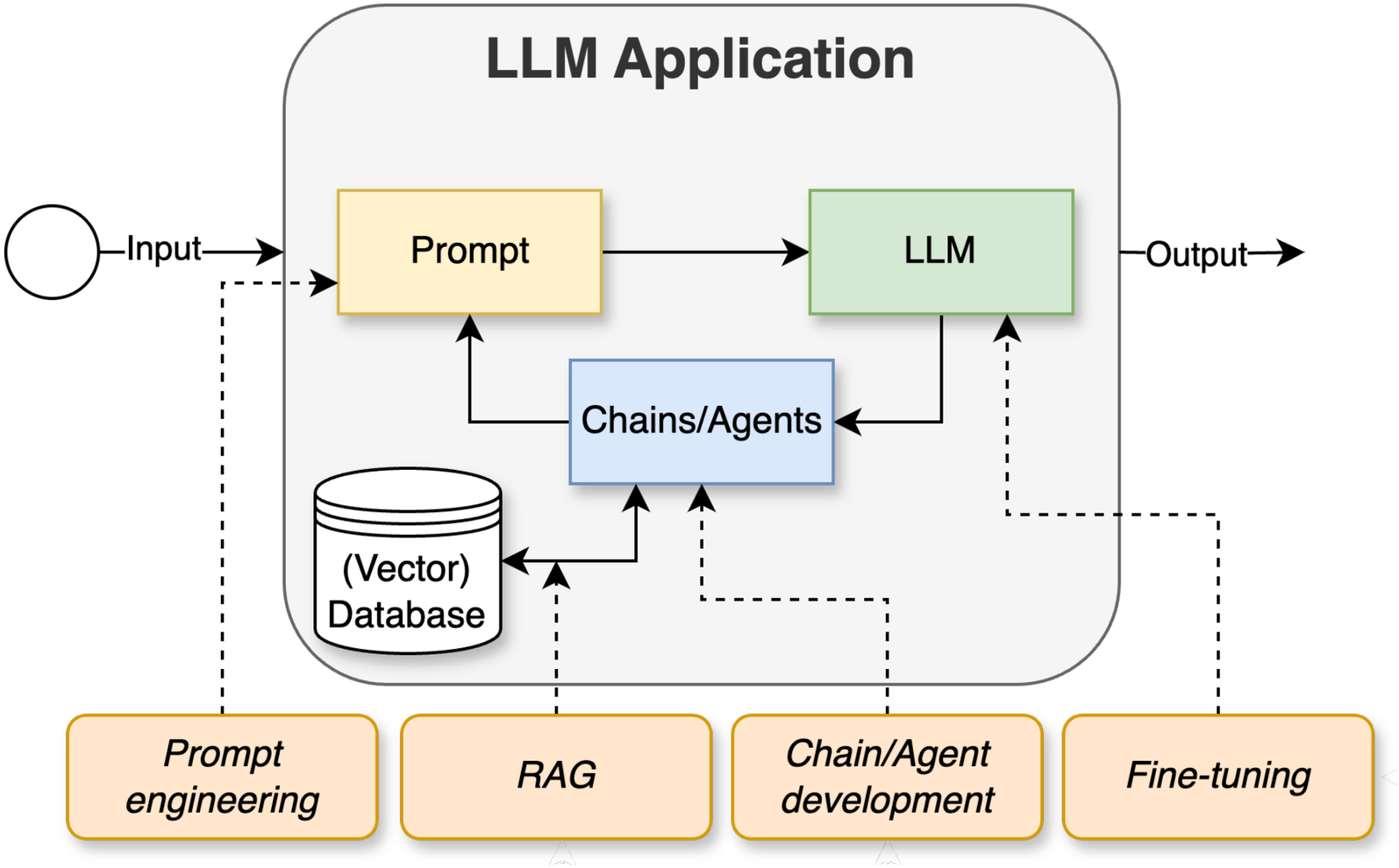
Output characteristics:

- Bias
- ☠ Toxicity
- Helpfulness

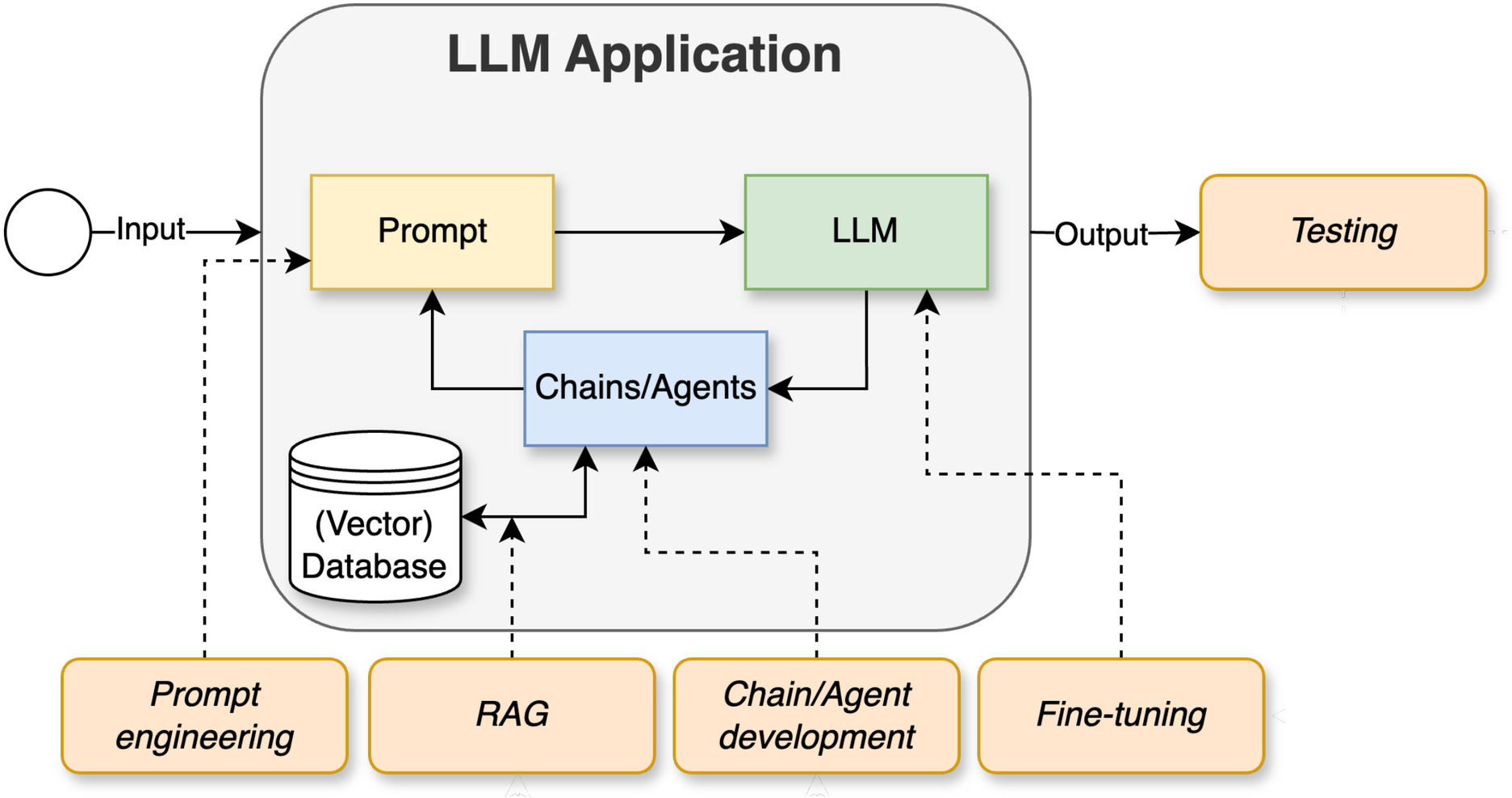
Operational characteristics:

- Latency
- Total incurred cost
- Memory usage

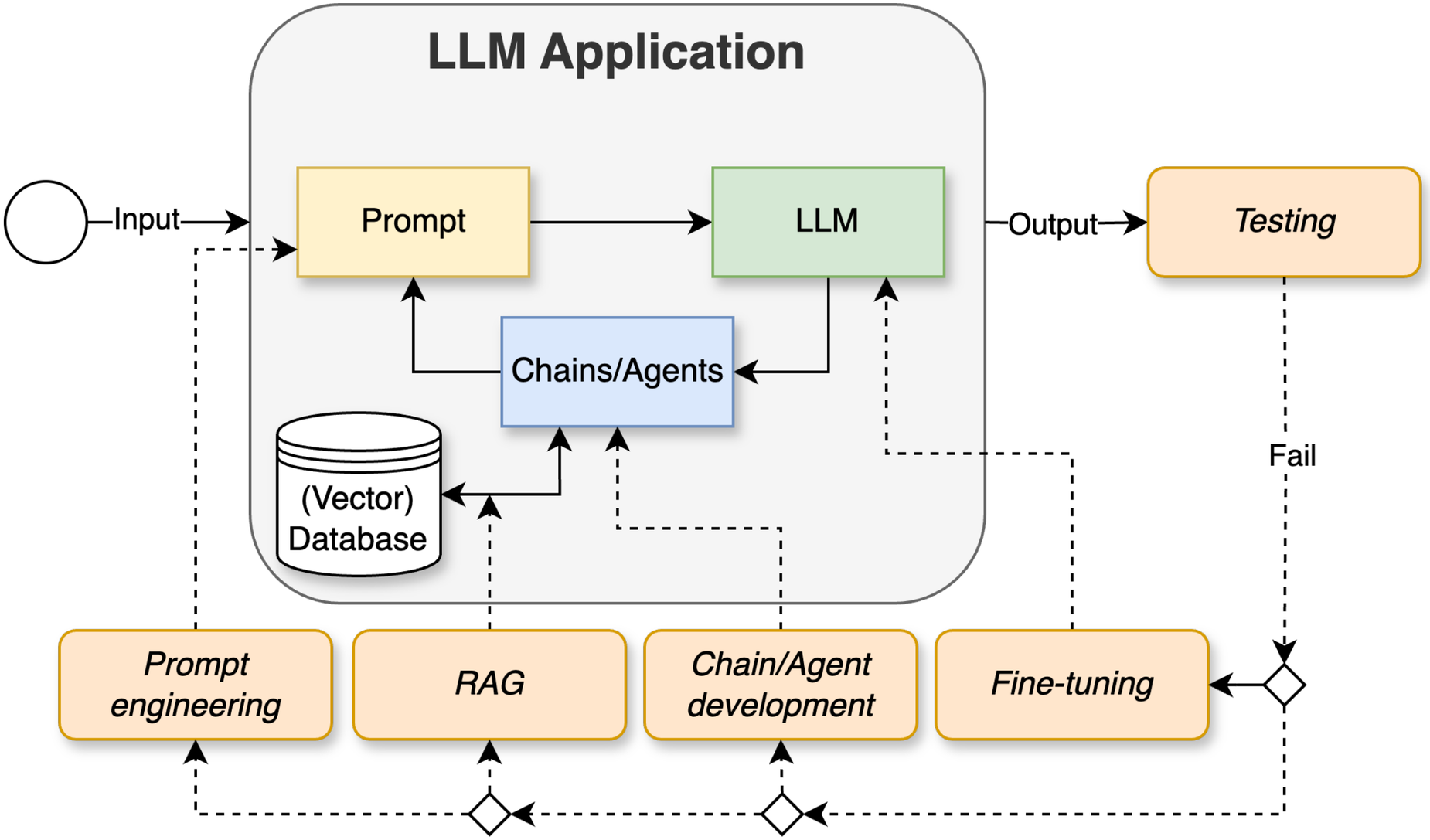
The development cycle



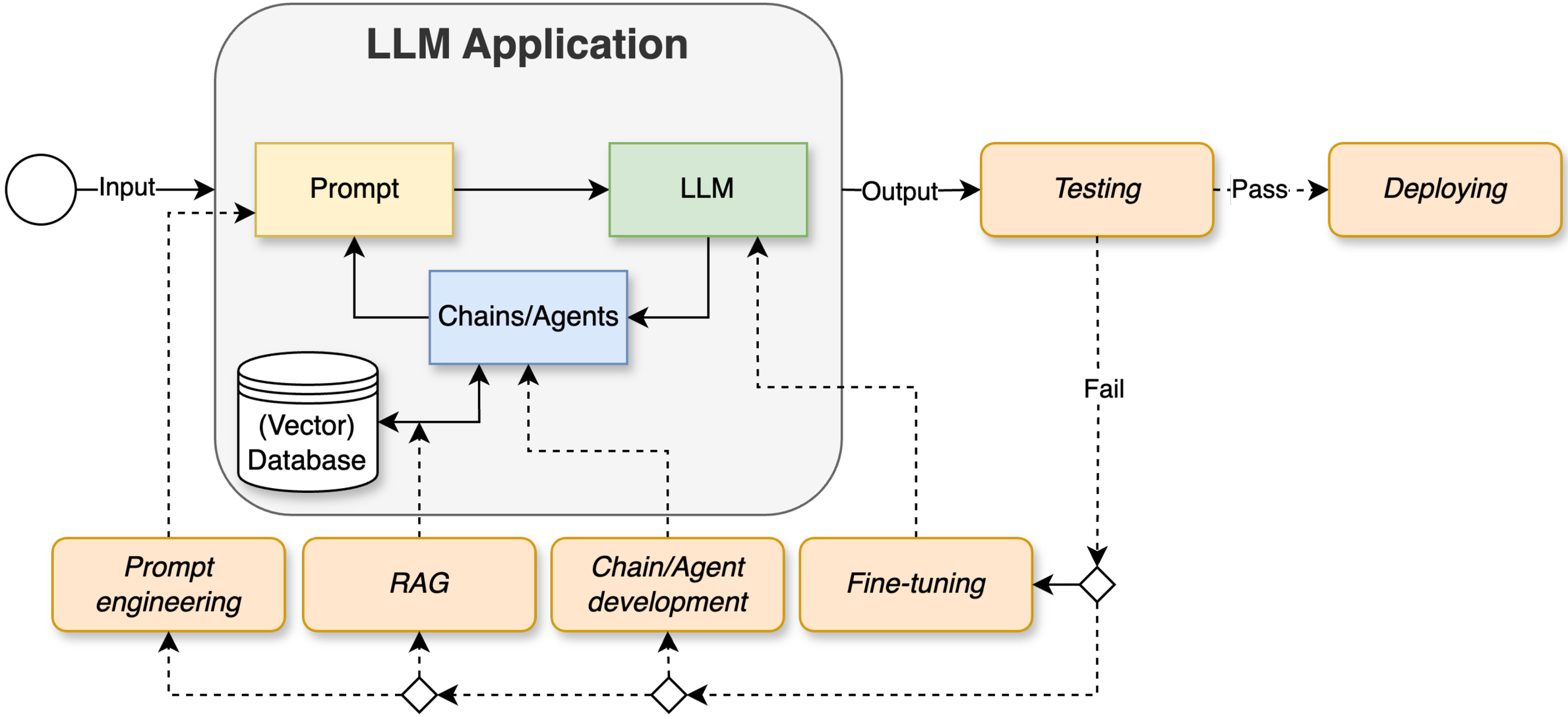
The development cycle



The development cycle



The development cycle



Let's practice!

LLMOPS CONCEPTS