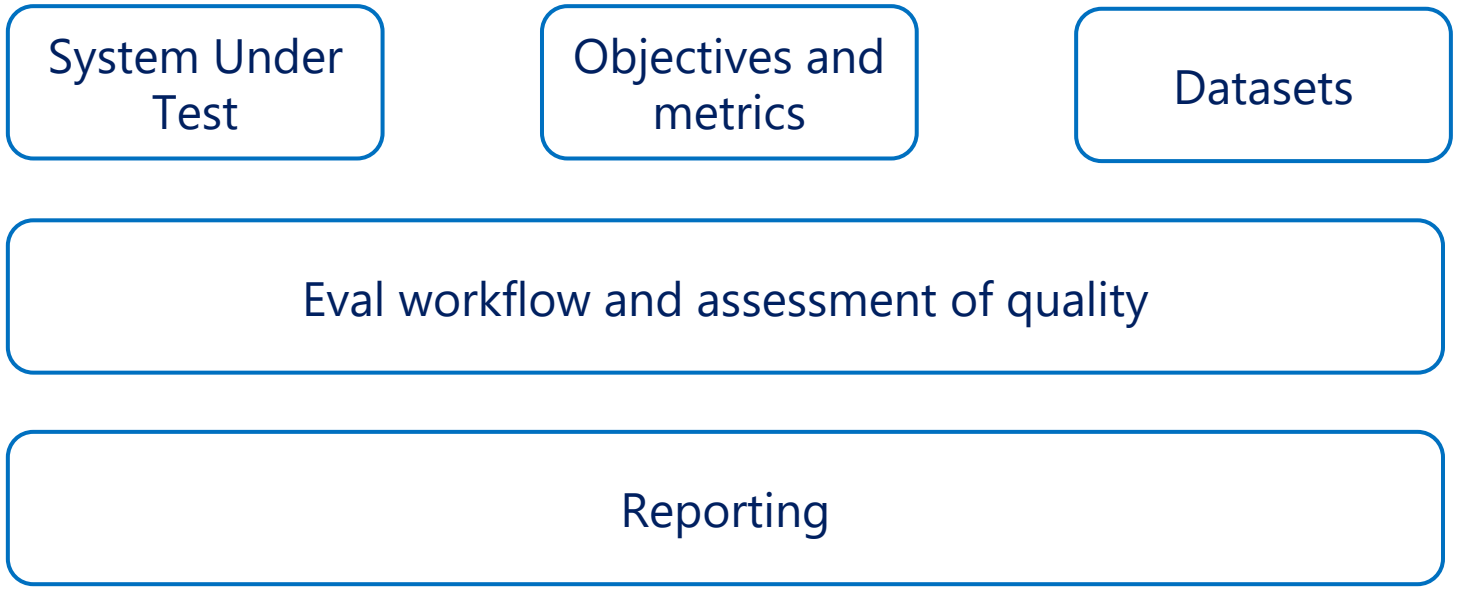
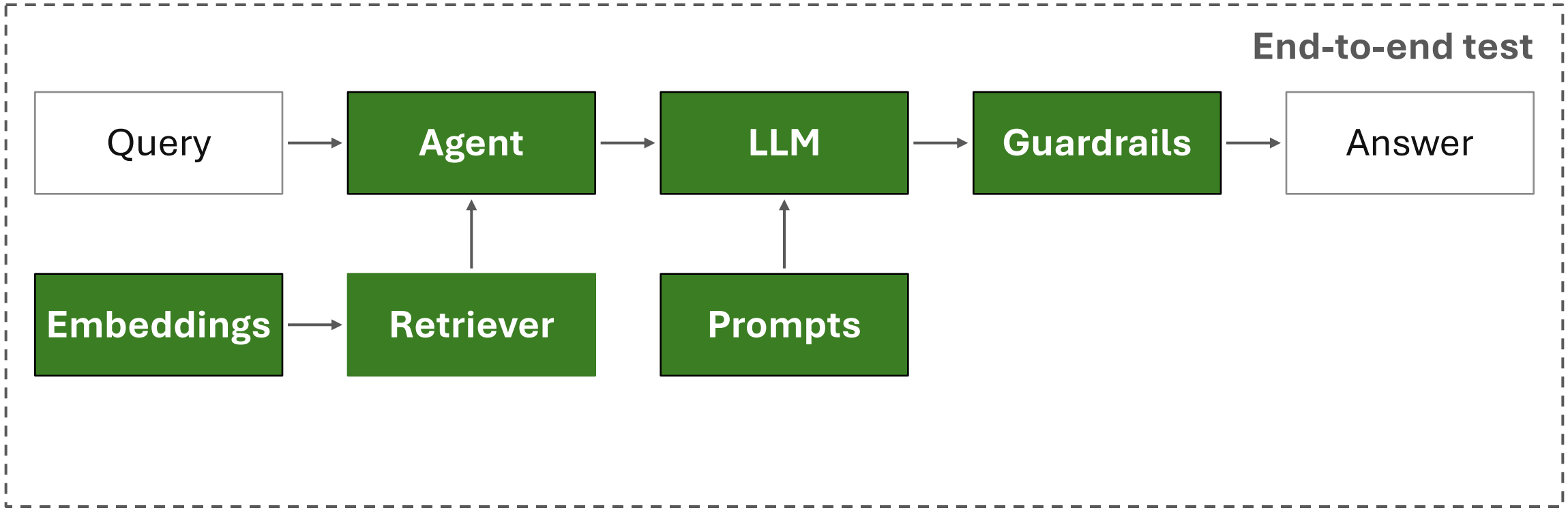
Getting started with Experiment Design: Objectives and System Under Test


System Under Test
Objectives and metrics
Datasets
Eval workflow and assessment of quality
Reporting

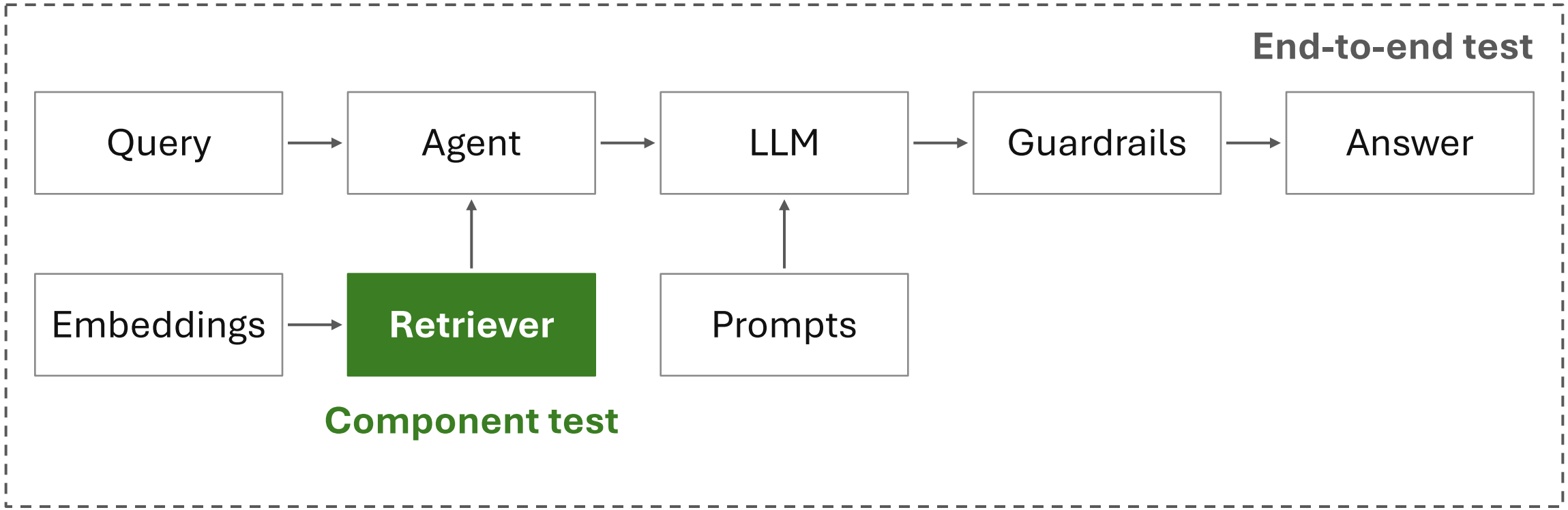
System under test

What are we testing – under what conditions?



System under test

What are we testing – under what conditions?



System under test / experimental conditions

System / Condition	Notes
LLMs	test different models
Embeddings	LM based, transformers, tf-idf,...
Specific algorithms	eg, for clustering, k-means vs hdbscan vs hybrid
Prompts	Endless variants
Few shot examples	different numbers and ways to provide examples
Harness	For long running tasks / agents
...	

The curse of dimensionality (one of its many instantiations)

Dimension	Options	Runs
LLM	3 models	3
Prompt	4 variants	× 4 = 12
Few-shot examples	3 sets	× 3 = 36
Embeddings	2 models	× 2 = 72
Clustering algorithm	3 options	× 3 = 216
Datasets	5 datasets	**× 5 = 1,080**

Every dimension multiplies the number of runs. Fix most conditions to sensible defaults, vary one or two at a time, and only expand the grid where early results show it matters.

The key is to assess which ones can be evaluated independently

It also causes one more curse…. Multiple hypotheses testing

Goals of the experiments

Types of goals and RQs

Compare — V2 vs V1

Hypothesis and decision rule fixed up front.

Select — how good can one of N alternatives get?

No hypothesis; a value function, constraints and a pick rule.

Explore — is there a viable solution at all?

Questions not yet precise; the output is hypotheses, not a decision.

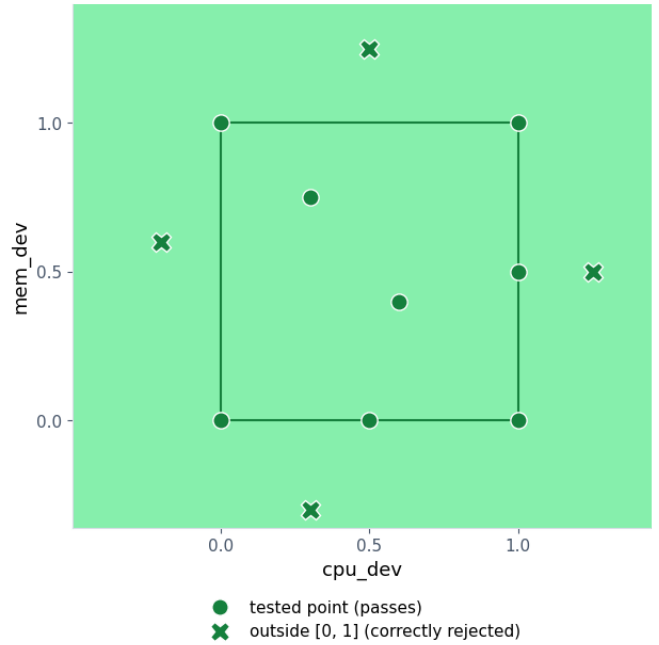
E1: Host health anomaly score in monitoring

```python
def deviation_score(cpu_dev: float, mem_dev: float) -> float:

    if not 0.0 <= cpu_dev <= 1.0:
        raise ValueError(f"cpu_dev must be in [0, 1], got {cpu_dev}")

    if not 0.0 <= mem_dev <= 1.0:
        raise ValueError(f"mem_dev must be in [0, 1], got {mem_dev}")

    return (cpu_dev**2 + mem_dev**2) ** 0.5
```

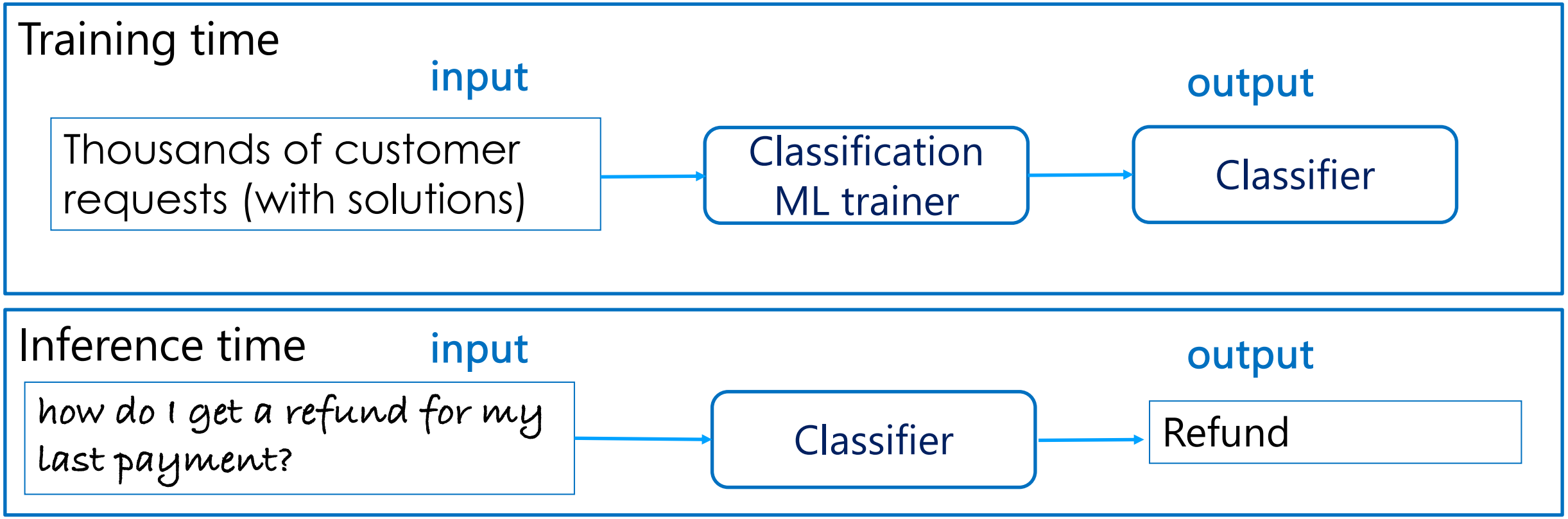


E1: Software 1.0 / function

Dimension	E1: host health anomaly score
System under test	The function
Goals of the experiment	Find bugs – gain confidence that the function responds correctly
Metrics, thresholds, confidence, baselines	Correctness (answer matches ground truth or raises the right exceptions) All test cases must pass **Baseline / comparison**: previous version.

E2: ML classifier for routing tickets to groups

A *classification* task: classify into ONE of these 5 classes:
account, **order**, **refund**, **payment**, **delivery**.

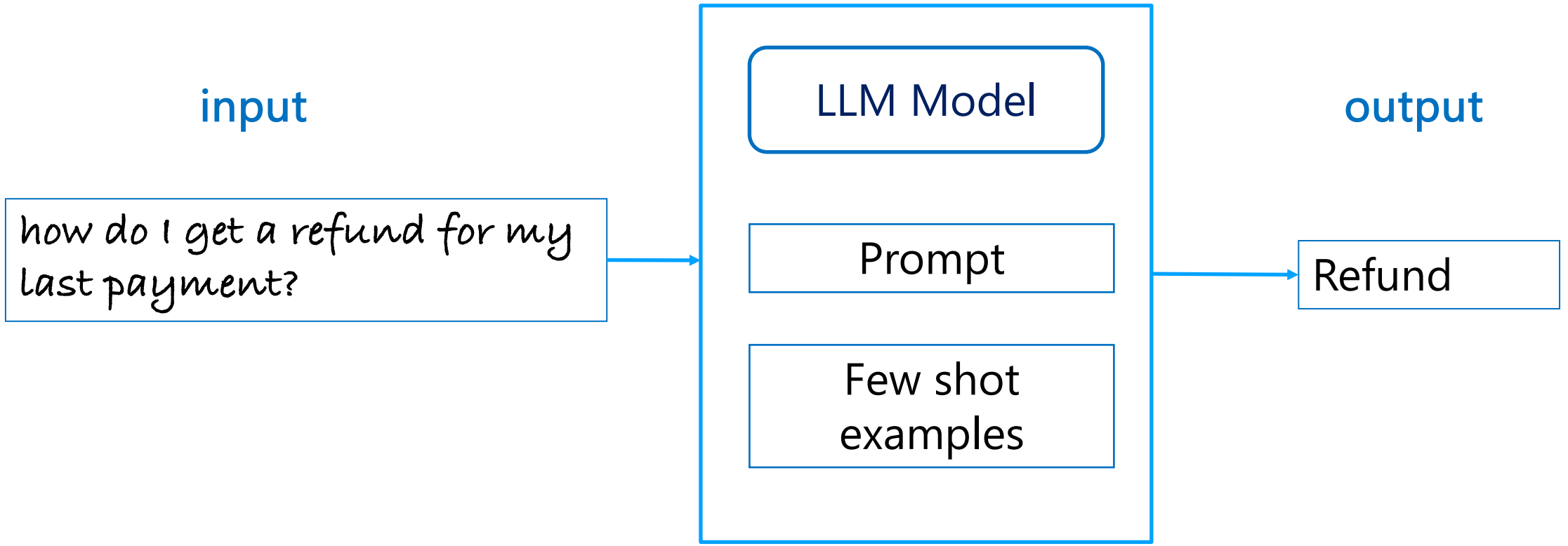


Trained classifier (E2)

Dimension	E2: ML ticket-routing classifier
System under test	The trained routing classifier (training pipeline + inference). End to end pipeline
Goals of the experiment	Route tickets to the right group. Decide whether classifier V2 is better than V1 and if we achieve the desired target (as outlined below)
Metrics, thresholds, confidence, baselines	Accuracy and per-class precision / recall. Inference latency **Target**: ≥90% overall accuracy and no class below 80% recall. **Comparison**: Previous model version (V1). V2 vs V1 on accuracy with 95% probability

E3: GenAI, few shot classifier for routing tickets to groups

A *classification* task: classify into ONE of these 5 classes:
account, **order**, **refund**, **payment**, **delivery**.



GenAI / few-shot LLM classifier (E3)

Dimension	E3: few-shot LLM ticket classifier
System under test	Eg, 3 versions of the prompt + 2 versions of few shots, fixed model.
Goals of the experiment	Can a few-shot LLM match the trained classifier's routing quality, without labeled training data? Which version achieves it? How good does it get?
Metrics, thresholds, confidence, baselines	Same metrics as E2.

E4: self-solve via clustering

Training time

input

Thousands of customer requests (with solutions)

Clustering algorithm

Solution extractor

Output:

1. Clustering of issues
2. Solution per cluster

Inference time

how do I get a refund for my last payment?

Cluster matching

Answer generator

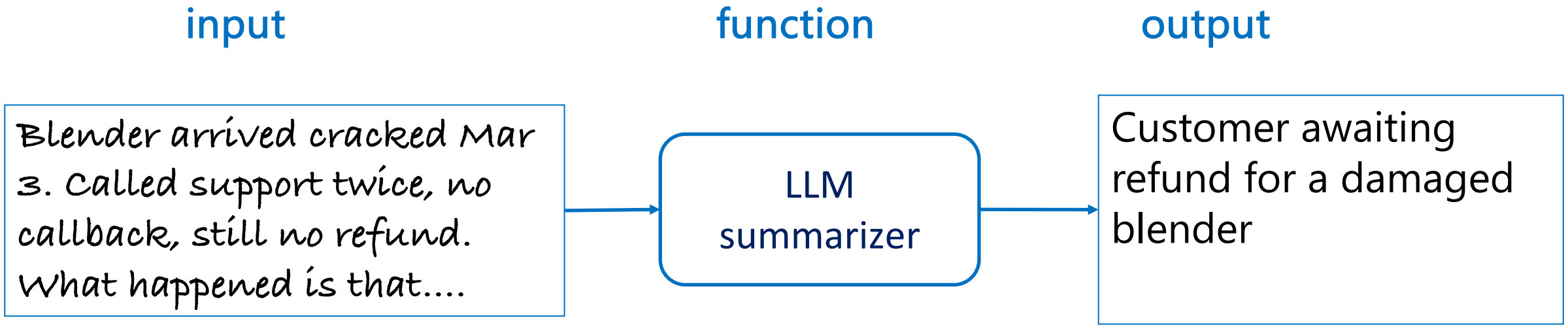
output

Apply at xyz.com

Auto-resolution(E4)

Dimension	E4: self-solve via clustering
System under test	Clustering component only. Experimental conditions include 2 clustering algo (3 hyperparams settings for each), 3 different text columns
Goals of the experiment	Which clustering algo leads to better results?
Metrics, thresholds, confidence, baselines	E2E auto-resolve: **coverage** (share of cases auto-resolved) and **precision** (share of those resolved correctly) **Target**: Coverage up by ≥5 pts at equal or better precision. 95% prob. **Baseline**: *Berttopic* with default settings

E5: text summarization



GenAI / text summarization (E5)

Dimension	E5: text summarization
System under test	Test 4 prompts x 2 LLMs over case summarization
Goals of the experiment	Pick the best prompt and estimate its effectiveness with respect to V1
Metrics, thresholds, confidence, baselines	**Faithfulness** (no invented facts). **Completeness** (coverage of key points). **Conciseness**,.. **Constraints**: Objective: highest mean judge score (Faithfulness + completeness). Ties → cheaper prompt 95% prob of improvement. Else keep current **Baseline**: Current basic prompt and model

Metrics
(and thresholds, baselines, uncertainty)

What does "***good***" look like?

Quality
scorecard

(problem and
domain specific)

Latency
Cost

Security
Privacy

Consistency
Robustness
"Predictability"

And again we can suffer from multiple hypotheses testing