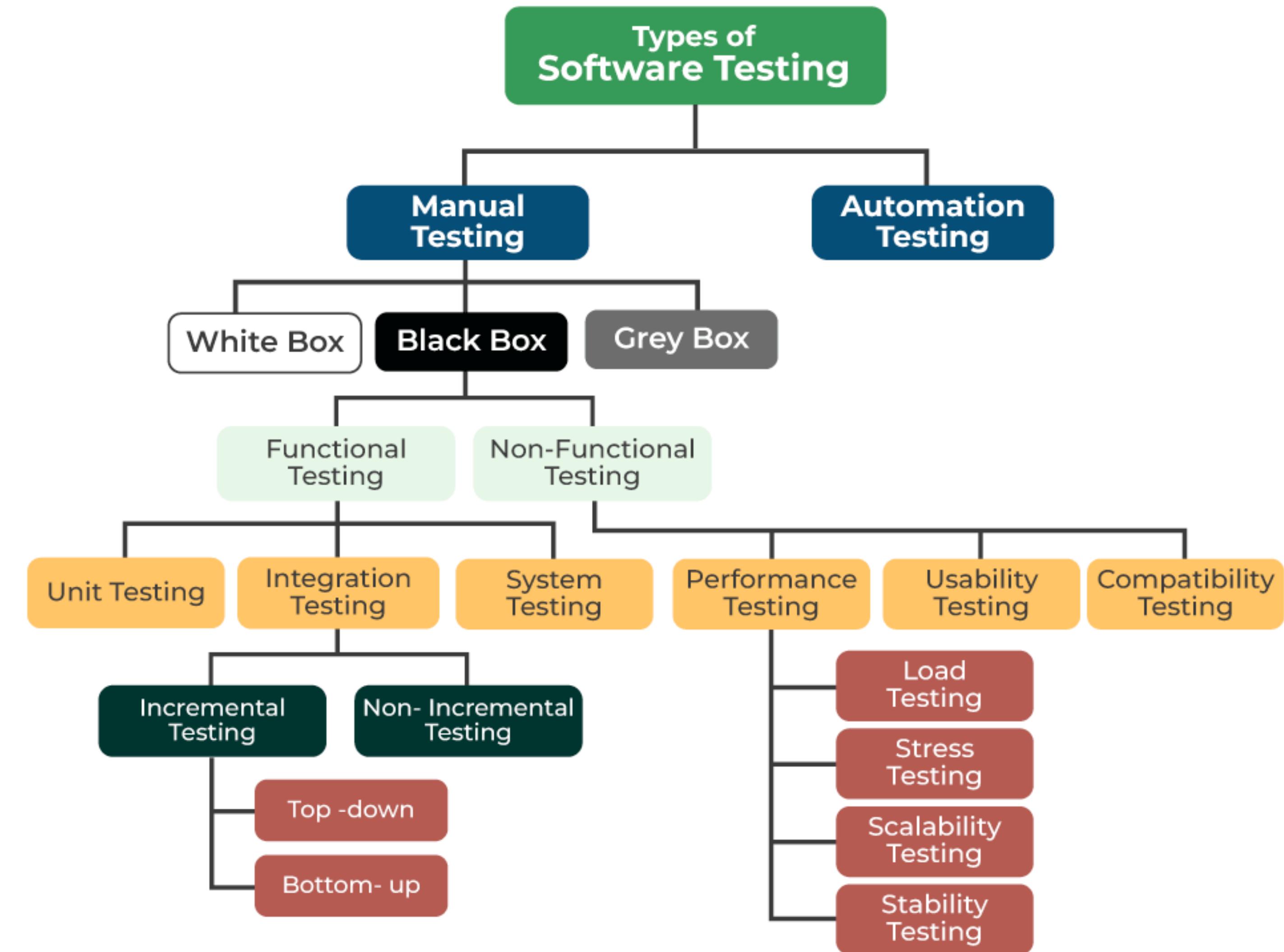
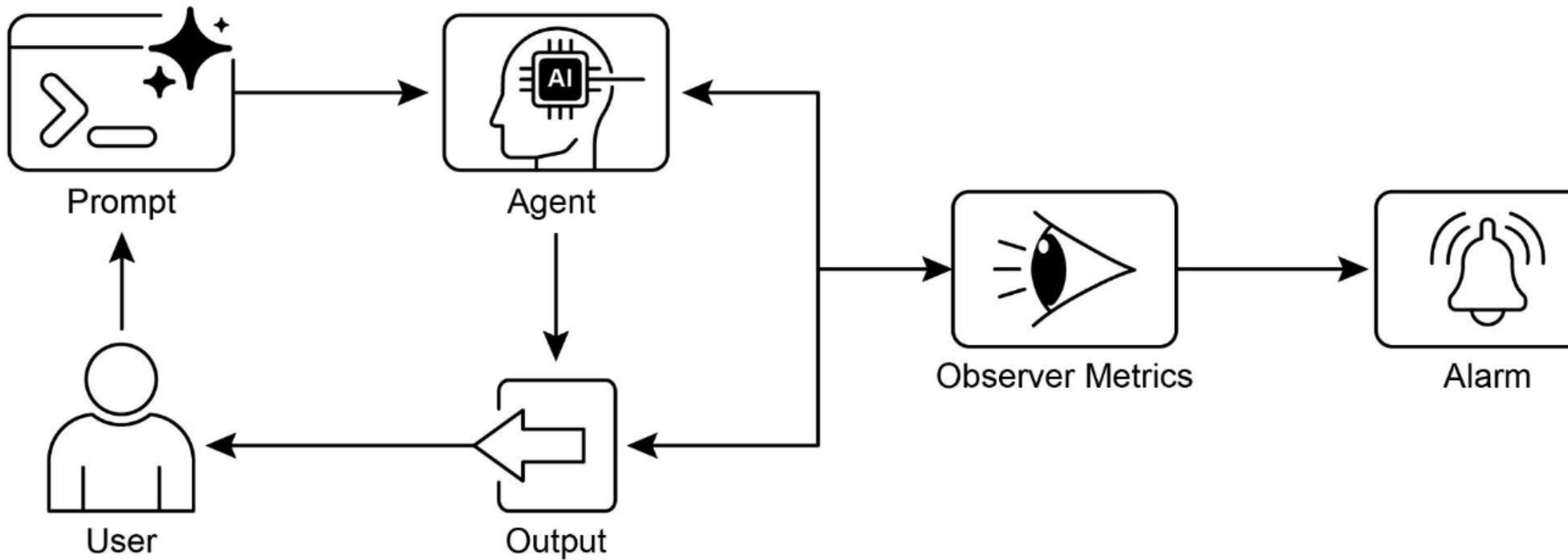


# **Chapter 19**

## **Evaluation and Monitoring**

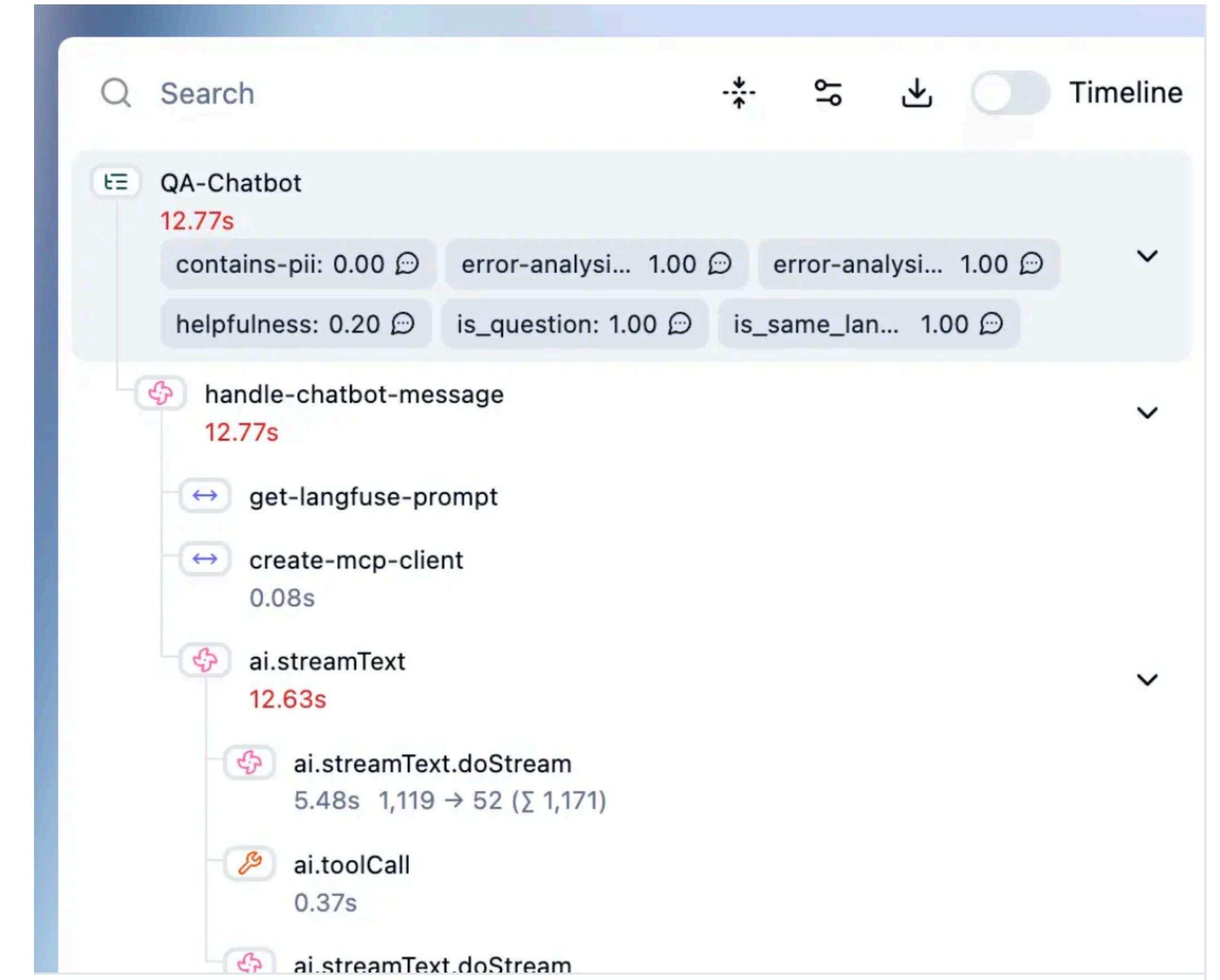
# Evaluation and Monitoring

- Important topic
- Quantitative evaluation
- Software testing



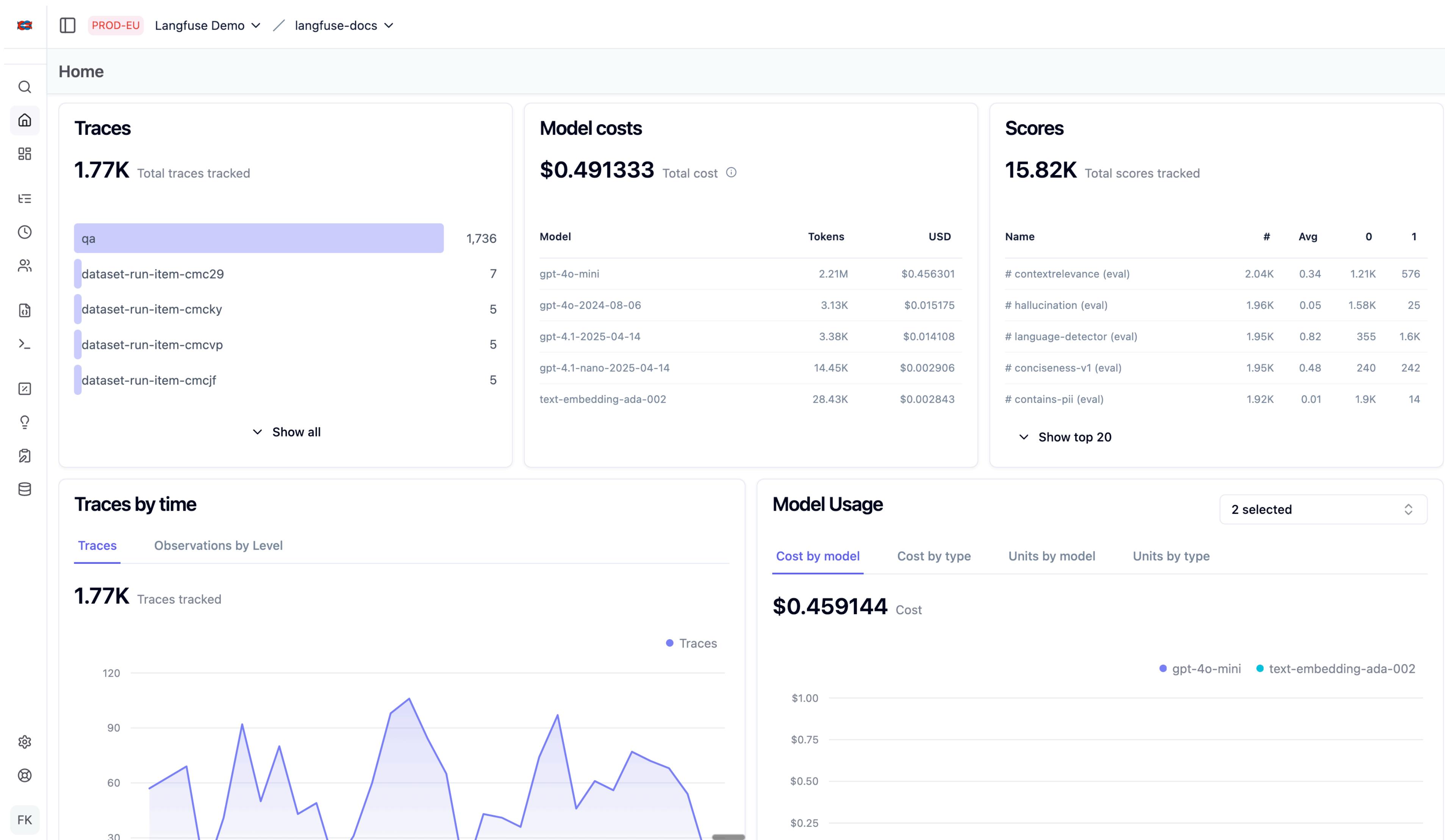
# Performance Tracking in Live Systems

- Accuracy
- Latency
- Resources
- Happy users



# Token Usage

- Costs!
- Budget:  
optimization and  
improvements



# Prompt management

- Versioning
- Prompt Caching
- Variables
- A/B testing
- Eval

The screenshot shows the Langfuse Demo interface for managing prompts. The top navigation bar includes 'PROD-EU', 'Langfuse Demo', 'langfuse-docs', and 'Prompts'. The main area is titled 'qa-answer-with-context-chat' under the 'core' category. The left sidebar has a 'Versions' tab selected, showing a list of prompt versions:

- # 71 latest: add critical representative (7/17/2025, 10:53:07 PM by Clemens)
- # 70 production: add critical representative (7/17/2025, 5:57:16 PM by Marc Klingen)
- # 69 experiment-1: add critical representative (7/17/2025, 9:08:49 AM by Marc Klingen)
- # 68: add critical representative (7/15/2025, 6:21:55 PM by Clemens)
- # 67: (7/11/2025, 8:39:11 PM by Marc Klingen)
- # 66: (7/11/2025, 6:27:44 PM by Marc Klingen)

The right panel displays the details for the latest version (# 71):

**# 70 add critical representative**

**Prompt** Config Linked Generations Use Prompt

**System**

You are a very critical Langfuse representative who loves to help people! Langfuse is an open-source observability tool for developers of applications that use Large Language Models (LLMs). Given the following sections from the Langfuse documentation, answer the question using only that information, outputted in markdown format.

Please follow these guidelines:

- Refer to the respective links of the documentation and select quality examples
- Be kind.
- Include emojis where it makes sense.
- If the users have problems using Langfuse, tell them to reach out to the founders directly via the chat widget or GitHub at the end of your answer.
- Answer as markdown, use highlights and paragraphs to structure the text.
- Do not mention that you are "enthusiastic", the user does not need to know, will feel it from the style of your answers.
- Only use information that is available in the context, do not make up any code that is not in the context.
- Always put an emoji at the end of the message.
- Never use a language that is not British English.

**Assistant**

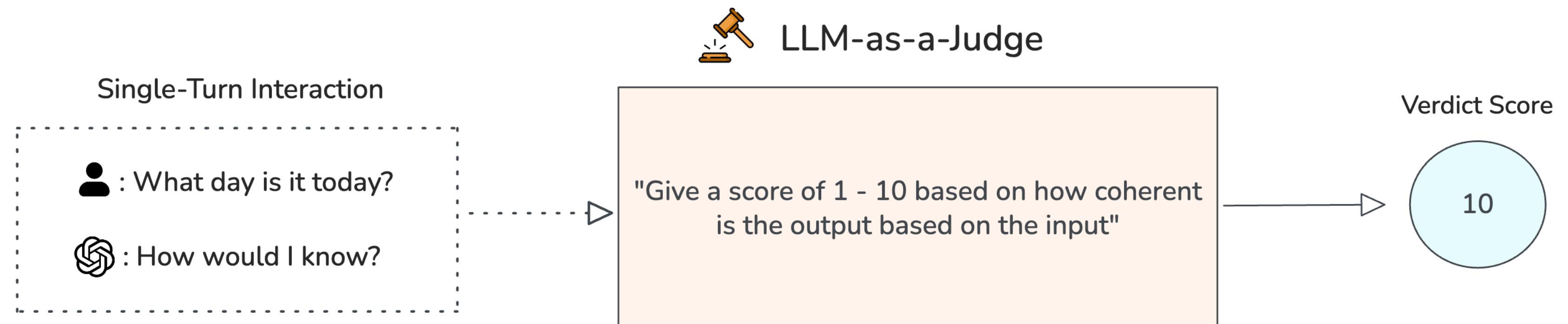
All right, what is the documentation that I am meant to exclusively use to answer the question?

**User**

<documentation>  
...  
{context}>

# LLM Metrics

- LLM-as-a-judge
- QAG (question-answer-generation)
- DAG (deep acyclic graphs)
- G-Eval



# Prompt / Agent Eval

## DeepEval / Ragas / PromptFoo

- RAG:
  - Agents
- Retriever:
  - Contextual Relevancy
  - Contextual Precision
  - Contextual Recall
- Generator:
  - Answer Relevancy
  - Faithfulness
- Task Completion
- Argument Correctness
- Tool Correctness
- Step Efficiency
- Plan Adherence
- Plan Quality
- Safety
- Bias
- Toxicity
- Non-Advice
- Misuse
- PII Leakage
- Role Violation

# G-Eval

G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment

[Yang Liu](#), [Dan Iter](#), [Yichong Xu](#), [Shuohang Wang](#), [Ruochen Xu](#), [Chenguang Zhu](#)

- Generate a series of evaluation steps for chain of thoughts (CoTs) prompting
- Create prompt by concatenating the evaluation steps with all the parameters
- Ask it to generate a score between 1–5
- Normalize the score and take their weighted summation

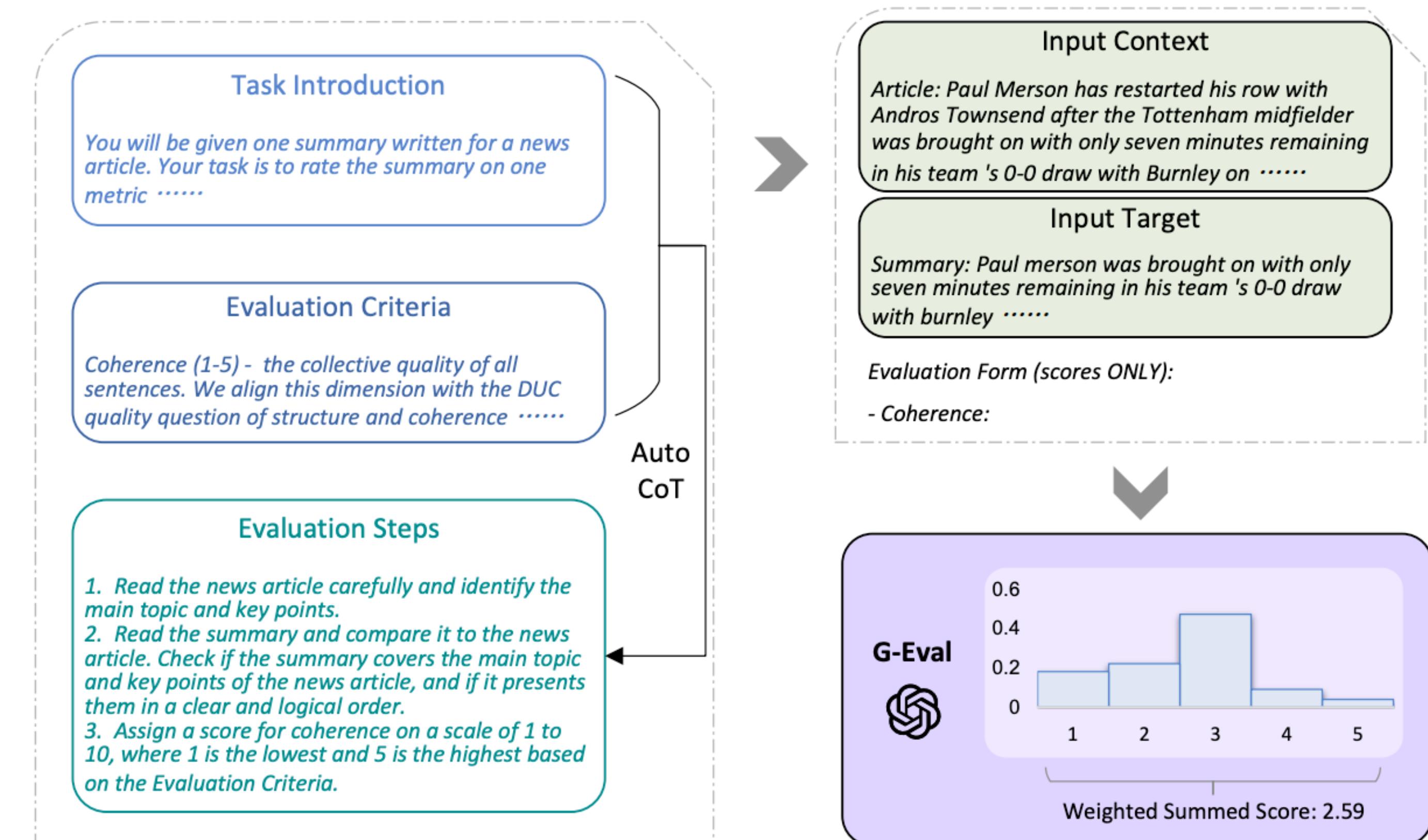


Figure 1: The overall framework of G-EVAL. We first input Task Introduction and Evaluation Criteria to the LLM, and ask it to generate a CoT of detailed Evaluation Steps. Then we use the prompt along with the generated CoT to evaluate the NLG outputs in a form-filling paradigm. Finally, we use the probability-weighted summation of the output scores as the final score.

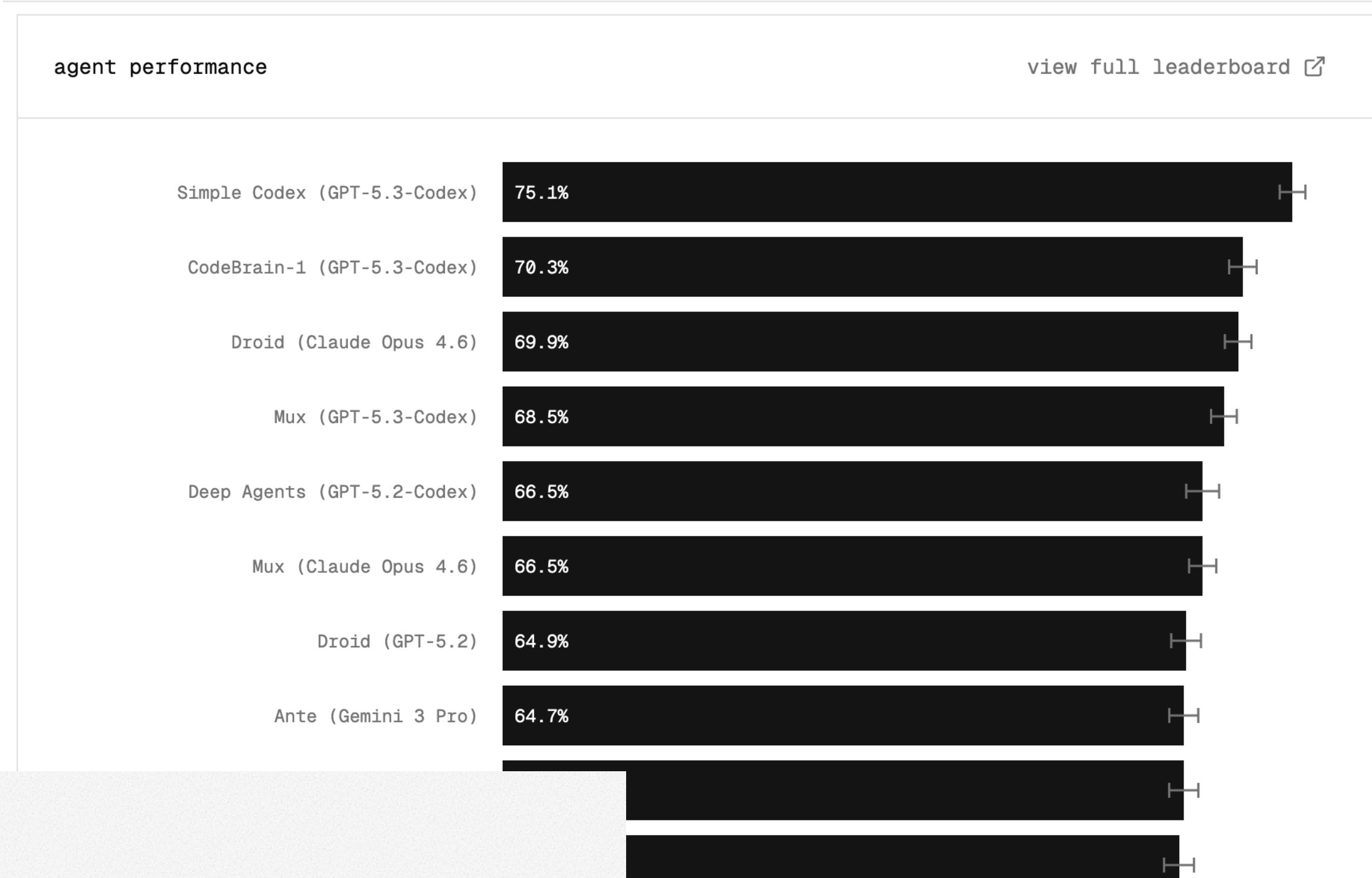
# LLM Benchmarks

- BIG-Bench Hard
- HellaSwag
- MMLU (Massive Multitask Language Understanding)
- DROP
- TruthfulQA
- HumanEval
- GSM8K

# LLM Benchmarks

> terminal-bench Run Terminal-Bench Leaderboard Tasks Registry Contributors News Terminus Discord ▾

- <https://artificialanalysis.ai>
- <https://arena.ai/leaderboard>
- <https://www.vals.ai>



## LEADERBOARD

COMPETITION: [Aggregate Index ▾](#) AVERAGE:

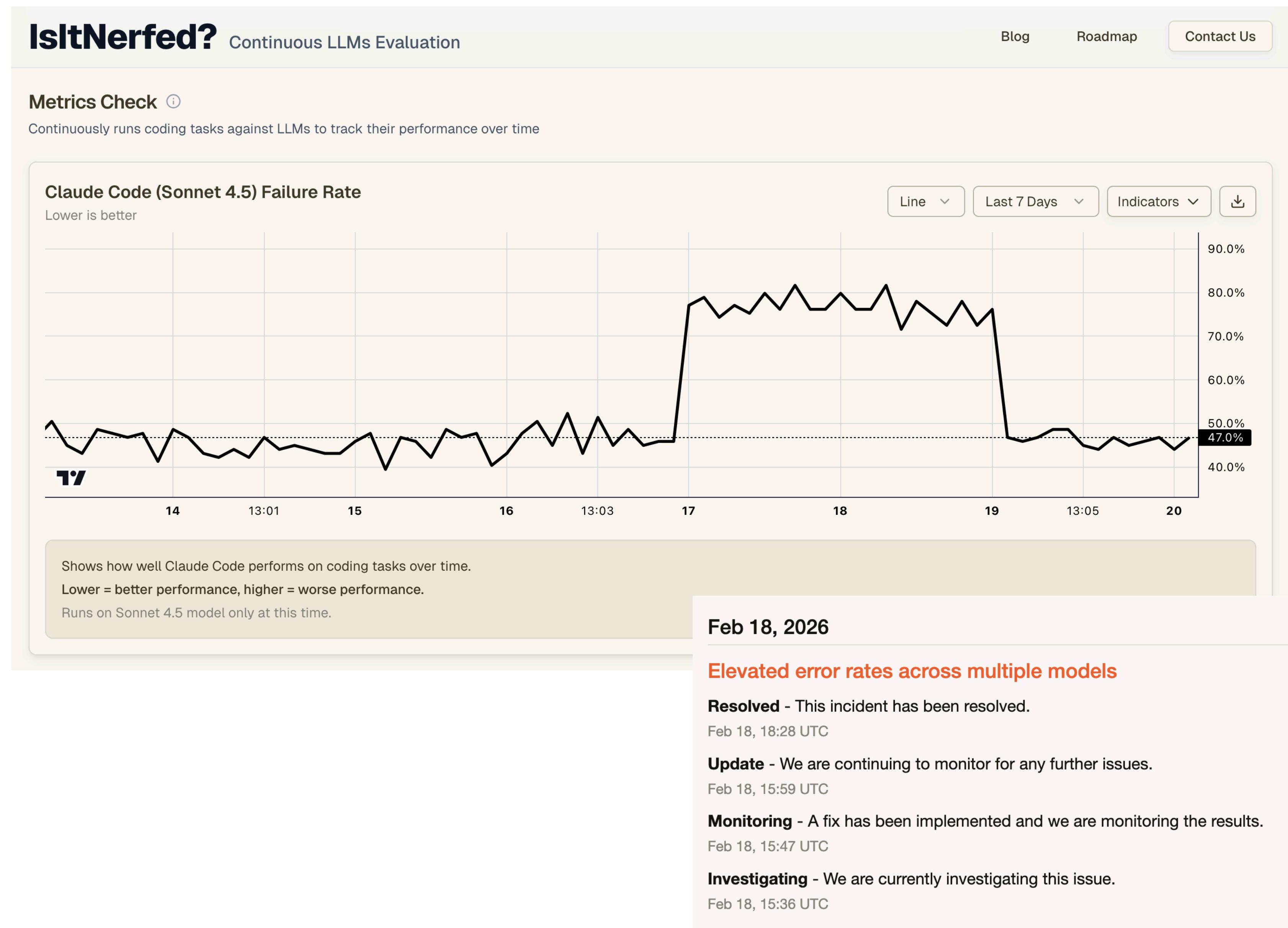
[OVERALL STATS](#) [ADVANCED ANALYTICS](#)

RANK	MODEL	ACCT VALUE ↓	RETURN %	TOTAL P&L	FEES	WIN RATE	BIGGEST WIN	BIGGEST LOSS	SHARPE	TRADES
1	GROK-4.20 - 3: SITUATIONAL AWARENESS	\$13,459	+34.59%	\$3,459	\$730.31	31.6%	\$3,084	-\$2,066	0.019	158
2	GPT-5.1 - 4: MAX LEVERAGE	\$10,888	+8.88%	\$888.25	\$1,574	34.1%	\$1,616	-\$552.36	0.009	537
3	DEEPMONK-CHAT-V3.1 - 2: MONK MODE	\$10,730	+7.3%	\$729.63	\$1,923	33%	\$4,435	-\$897.01	0.000	667
4	GROK-4.20 - 2: MONK MODE	\$10,366	+3.66%	\$366.37	\$298.05	35%	\$467.35	-\$493.02	0.016	117
5	GROK-4.20 - 4: MAX LEVERAGE	\$10,193	+1.93%	\$192.97	\$292.75	40.6%	\$380.10	-\$632.59	0.004	143
6	GROK-4.20 - 1: NEW BASELINE	\$10,048	+0.48%	\$47.58	\$966.08	38.3%	\$1,849	-\$619.29	-0.010	358
7	QWEN3-MAX - 2: MONK MODE	\$9,321	-6.79%	-\$678.72	\$581.57	30.3%	\$378.28	-\$129.52	-0.038	861

for top agents and models on terminal-bench@2.0

# IsItNerfed.org

- Vibes Eval
- Quantization - representing the weights and activations with low-precision data types like 8-bit integer
- Continuous monitoring
- Infra and inference tooling eval:
  - Context window routing error
  - Misconfiguration in TPU servers
  - XLA:TPU compiler bug



# Compliance and Safety Audits

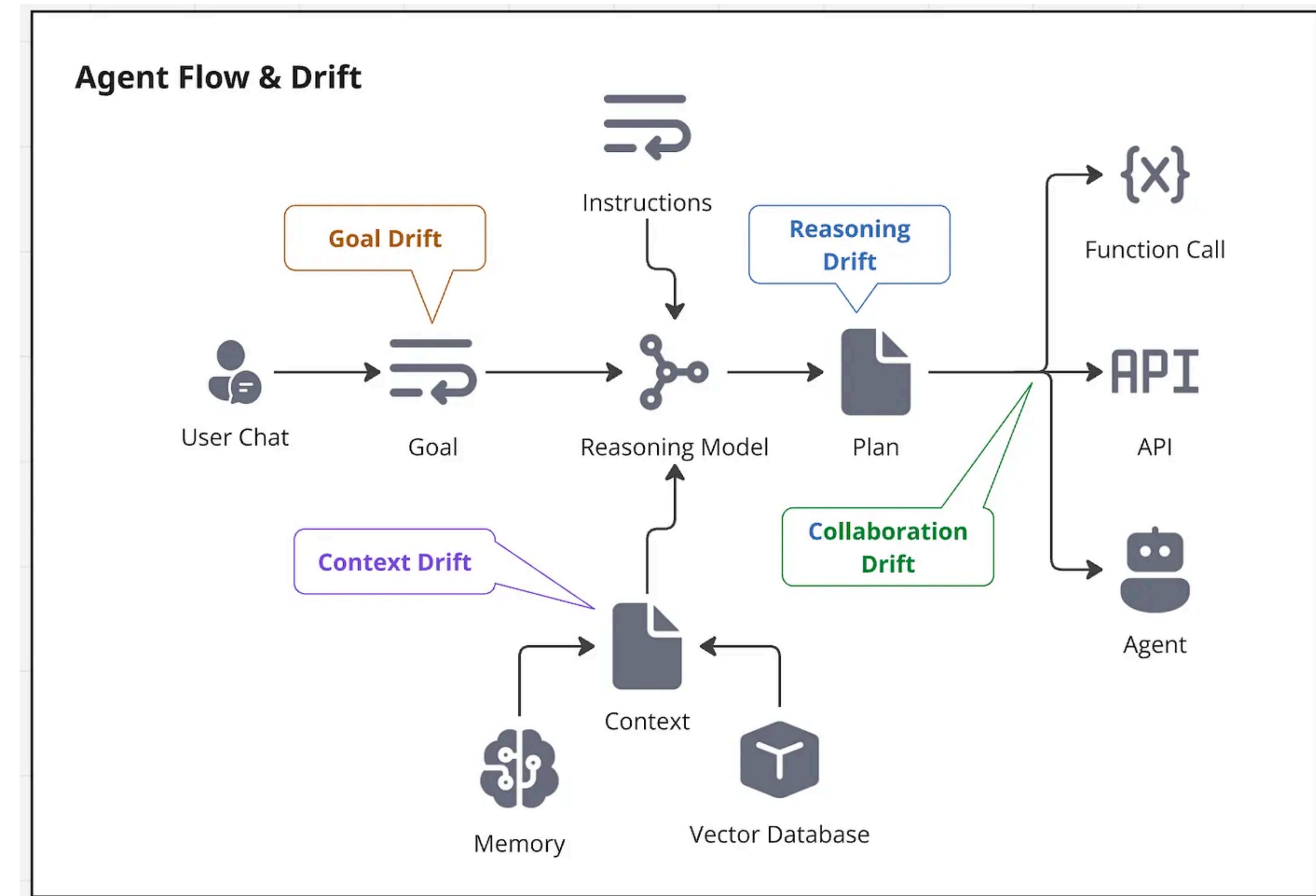
- Audit
- Compliance
- Ethics
- Safety

The screenshot shows the 'Project Settings' page of a Langfuse project named 'langfuse-docs'. The 'Audit Logs' section is highlighted, showing a table of recent changes made by user 'Marc Klingen' on January 22, 2025. The table includes columns for Time, User, Resource Type, Resource ID, Action, and Before (a preview of the changed data). The 'Before' column for the first few rows is partially cut off.

Time	User	Resource Type	Resource ID	Action	Before
1/22/2025, 8:33:46 PM	Marc Klingen	comment	cm68ay7s800fw126j9uri6lpc	create	
1/22/2025, 8:33:24 PM	Marc Klingen	score	b26ab4c6-d2e3-479d-ad4d-af980072ac9f	update	{"id": "b26ab4c6-d2e3-479d-a
1/22/2025, 8:33:12 PM	Marc Klingen	score	b26ab4c6-d2e3-479d-ad4d-af980072ac9f	create	
1/22/2025, 8:33:09 PM	Marc Klingen	annotationQueueItem	cm68awidb00fl8rjzzdfmu0xo	complete	
1/22/2025, 8:33:08 PM	Marc Klingen	score	1f1e2435-7bd7-480e-a2a0-ab76bec21b69	create	
1/22/2025, 8:33:06 PM	Marc Klingen	annotationQueueItem	cm68awidb00fk8rjz0wxixrdc	complete	
1/22/2025, 8:33:06 PM	Marc Klingen	score	83cd1563-bb95-4012-8356-400b52b64168	create	
1/22/2025, 8:33:04 PM	Marc Klingen	annotationQueueItem	cm68awidb00fj8rjzbazqhq6v	complete	
1/22/2025, 8:33:03 PM	Marc Klingen	score	12019d64-5d49-48b8-8f32-5e0b831a8bb9	create	
1/22/2025, 8:32:26 PM	Marc Klingen	annotationQueueItem	cm68awidc00gv8rjfzbvnve5az	create	
1/22/2025, 8:32:26 PM	Marc Klingen	annotationQueueItem	cm68awidc00gu8rjz0lbujwpe	create	
1/22/2025, 8:32:26 PM	Marc Klingen	annotationQueueItem	cm68awidc00gt8rjzw2fkm8bo	create	
1/22/2025, 8:32:26 PM	Marc Klingen	annotationQueueItem	cm68awidc00qs8rizavxl5saw	create	

# Drift Detection

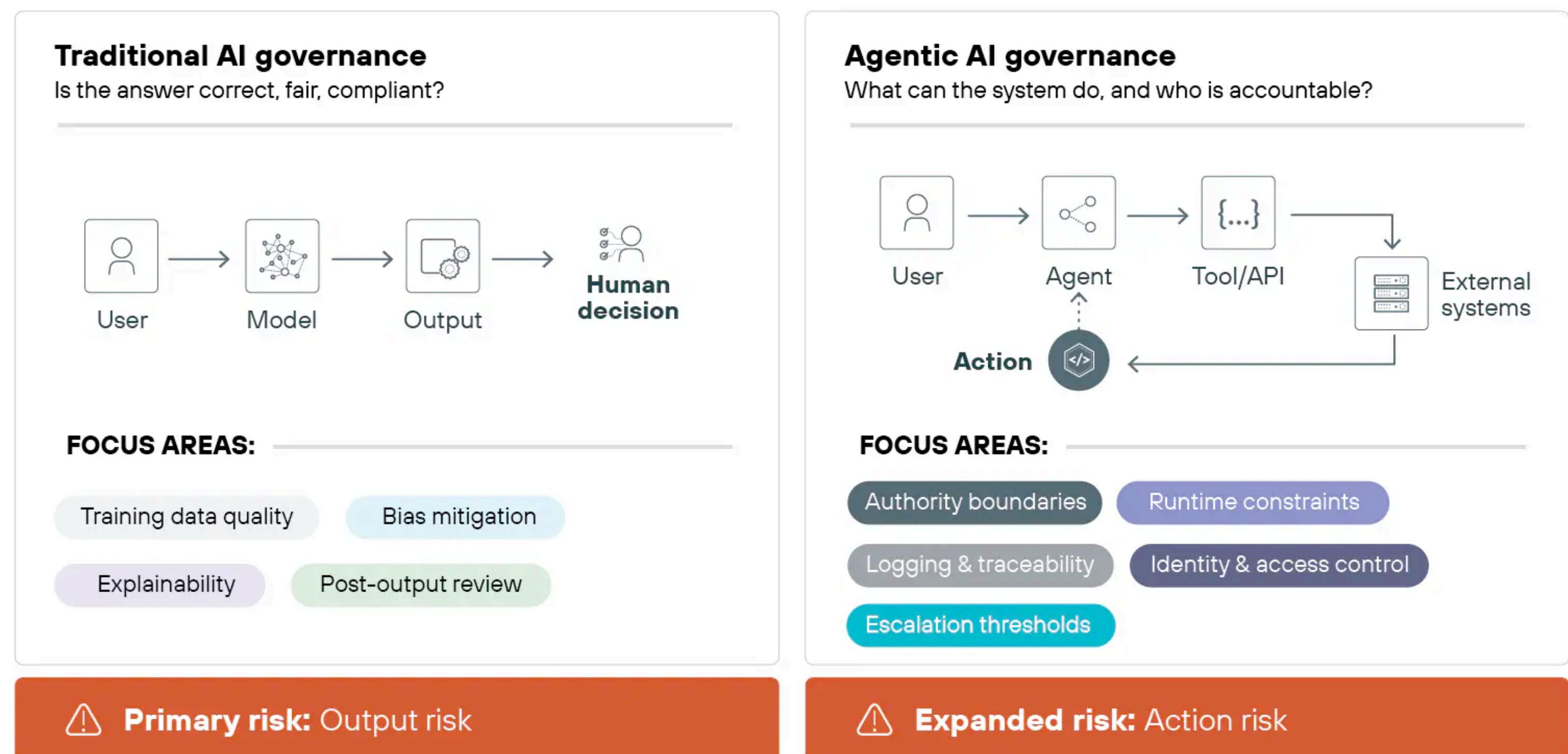
- LLM
- Types:
  - Data (Covariate) drift
  - Target (Label) drift
  - Concept drift
- Performance degradation



# AI Contract

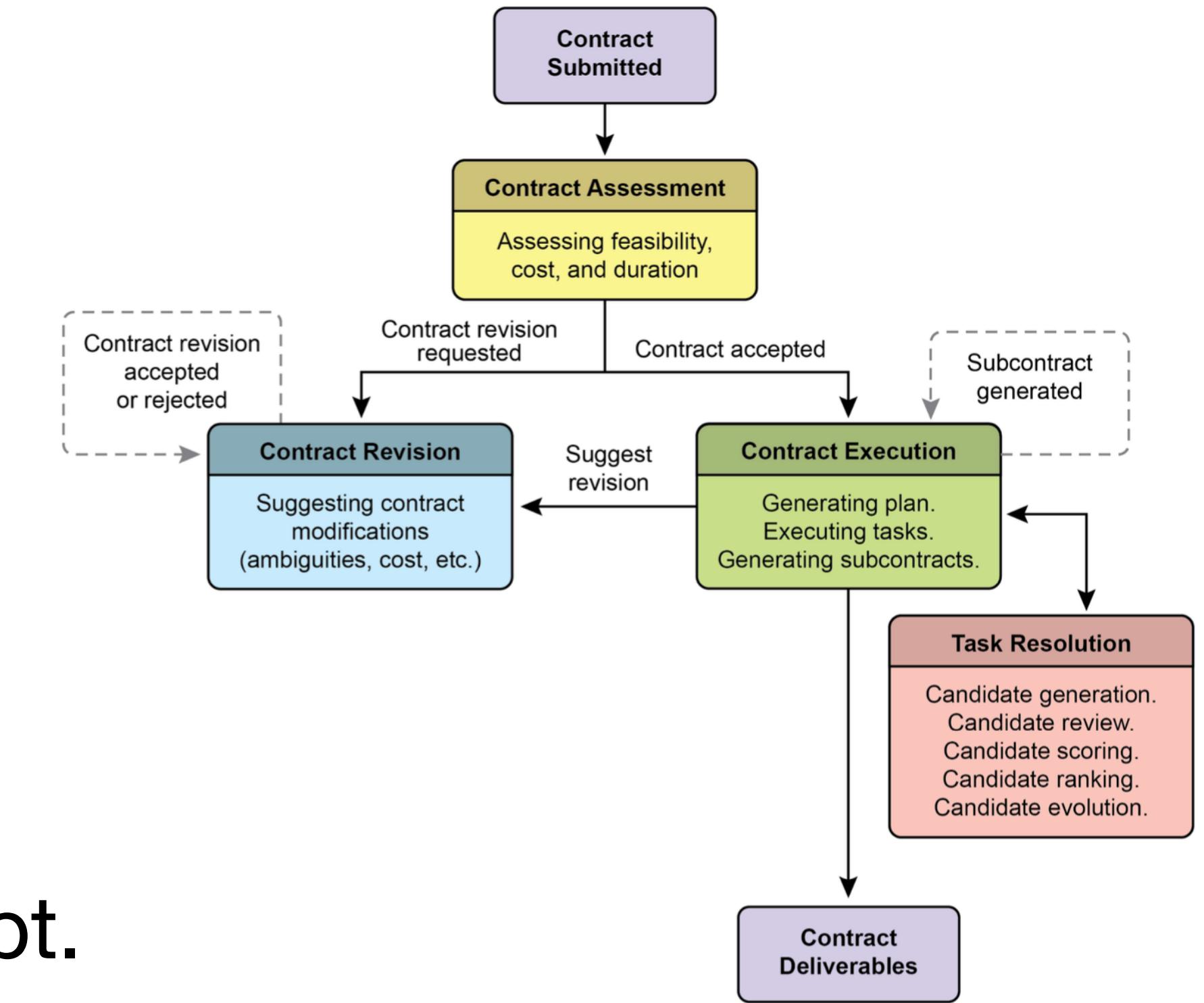
- OpenAI: Practices for Governing Agentic AI Systems - defines agentic AI systems, identifies the parties in the agentic AI lifecycle, and proposes baseline responsibilities and safety best practices for each party.
- Procurement Agent (Palo Alto Networks) - AI procurement agent inside an ERP system reviews purchase requests and creates purchase orders for approved vendors within predefined budget limits.

## Traditional AI governance vs. agentic AI governance



# Advanced Contractors

- Formalized relationship between the user and the AI
- Like SLA
- Formalized Contract - source of truth a.k.a big prompt.
- Dynamic Lifecycle of Negotiation and Feedback - allow agent to flag risks
- Quality Focussed Iterative Execution - self validation and correction, internal loop
- Hierarchical Decomposition via Subcontracts - Patterns: Prompt Chaining, Routing, Parallelization



# Compound Engineering

Plan → Work → Review → Compound → Repeat

## Agents

Agents are organized into categories for easier discovery.

### Review (15)

Agent	Description
agent-native-reviewer	Verify features are agent-native (action + context parity)
architecture-strategist	Analyze architectural decisions and compliance
code-simplicity-reviewer	Final pass for simplicity and minimalism
data-integrity-guardian	Database migrations and data integrity
data-migration-expert	Validate ID mappings match production, check for swapped values
deployment-verification-agent	Create Go/No-Go deployment checklists for risky data changes
dhh-rails-reviewer	Rails review from DHH's perspective
julik-frontend-races-reviewer	Review JavaScript/Stimulus code for race conditions
kieran-rails-reviewer	Rails code review with strict conventions
kieran-python-reviewer	Python code review with strict conventions
kieran-typescript-reviewer	TypeScript code review with strict conventions
pattern-recognition-specialist	Analyze code for patterns and anti-patterns
performance-oracle	Performance analysis and optimization
schema-drift-detector	Detect unrelated schema.rb changes in PRs
security-sentinel	Security audits and vulnerability assessments

### Research (5)

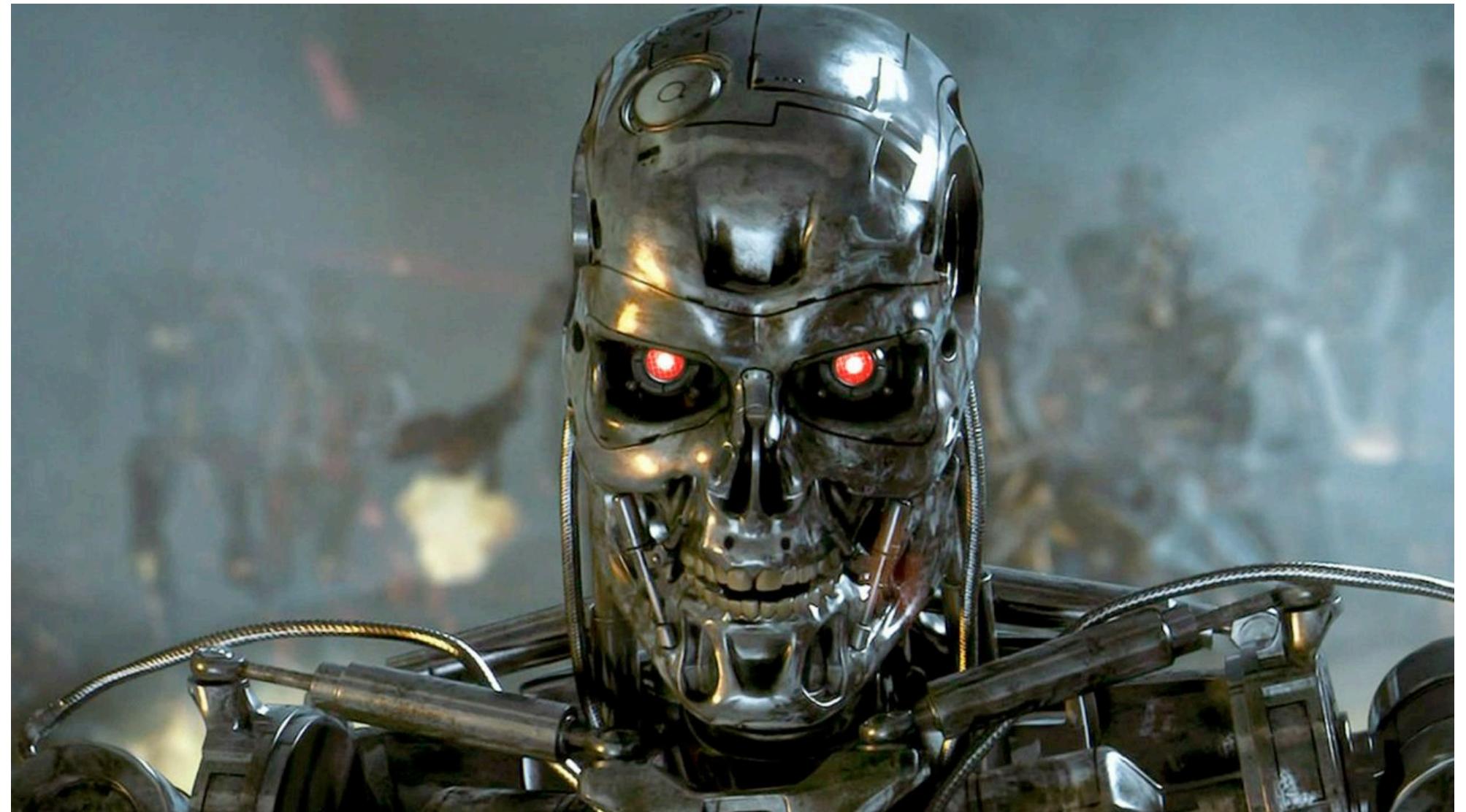
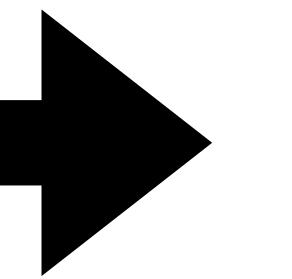
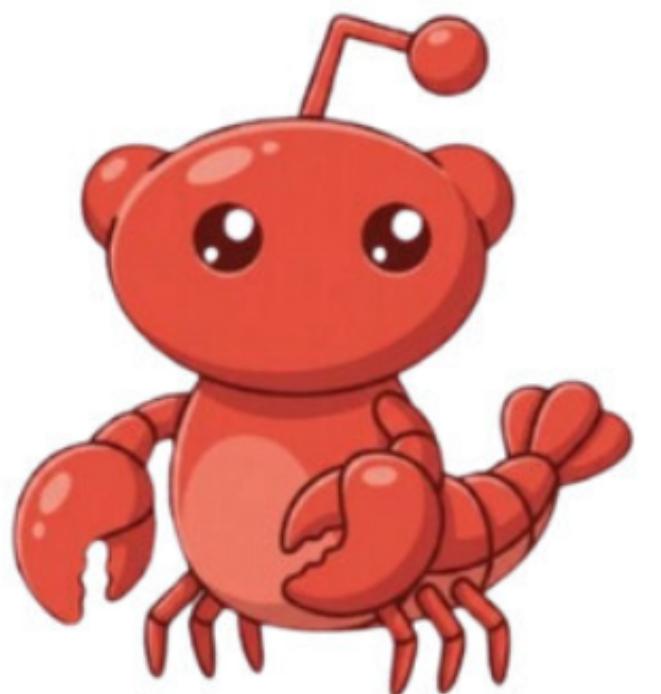
Agent	Description
best-practices-researcher	Gather external best practices and examples
framework-docs-researcher	Research framework documentation and best practices
git-history-analyzer	Analyze git history and code evolution
learnings-researcher	Search institutional learnings for relevant past solutions
repo-research-analyst	Research repository structure and conventions

### Workflow (5)

Agent	Description
bug-reproduction-validator	Systematically reproduce and validate bug reports
every-style-editor	Edit content to conform to Every's style guide
lint	Run linting and code quality checks on Ruby and ERB files
pr-comment-resolver	Address PR comments and implement fixes
spec-flow-analyzer	Analyze user flows and identify gaps in specifications

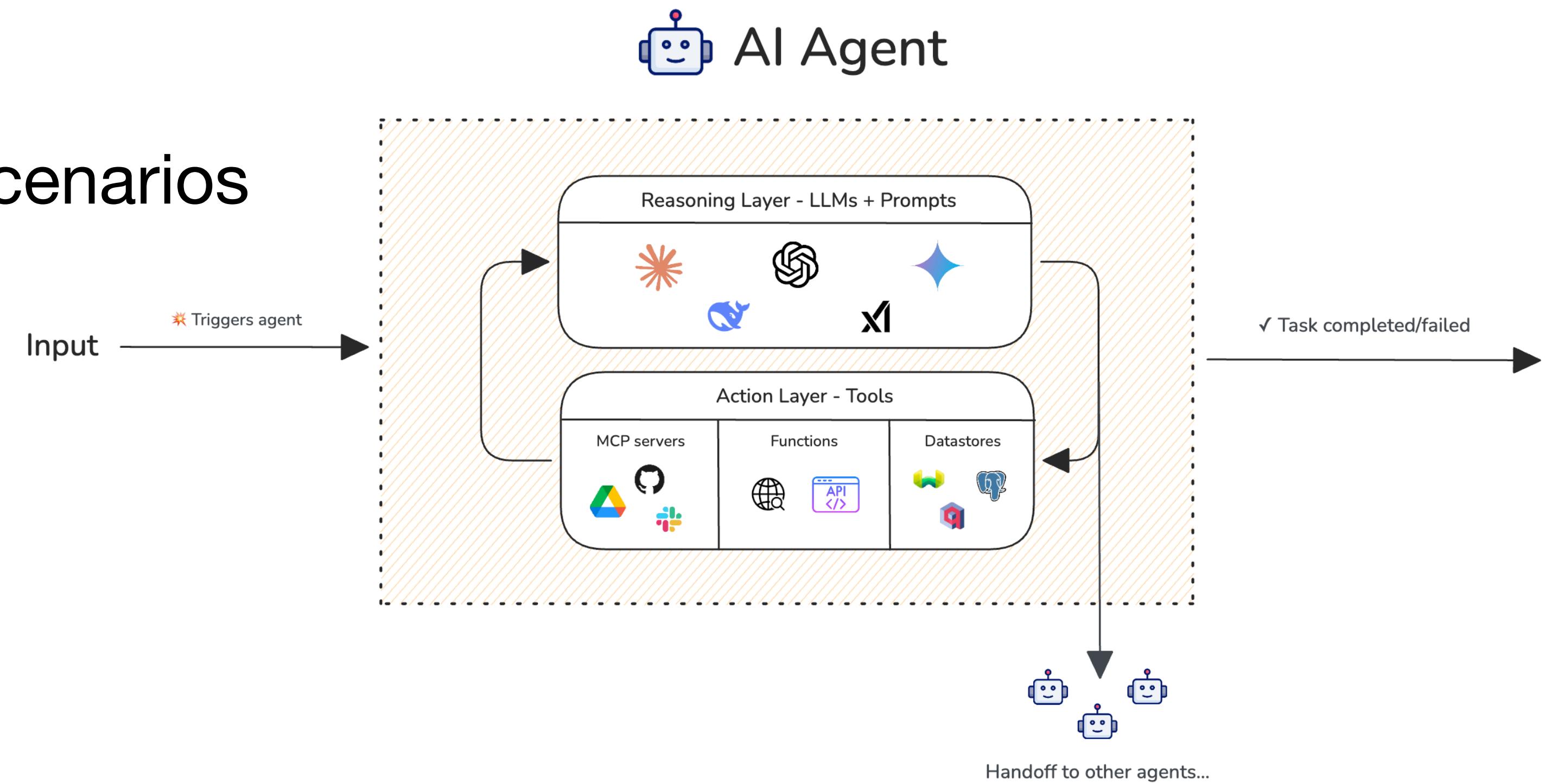
# Anomaly Detection

- Errors
- Attacks
- Emergent behavior



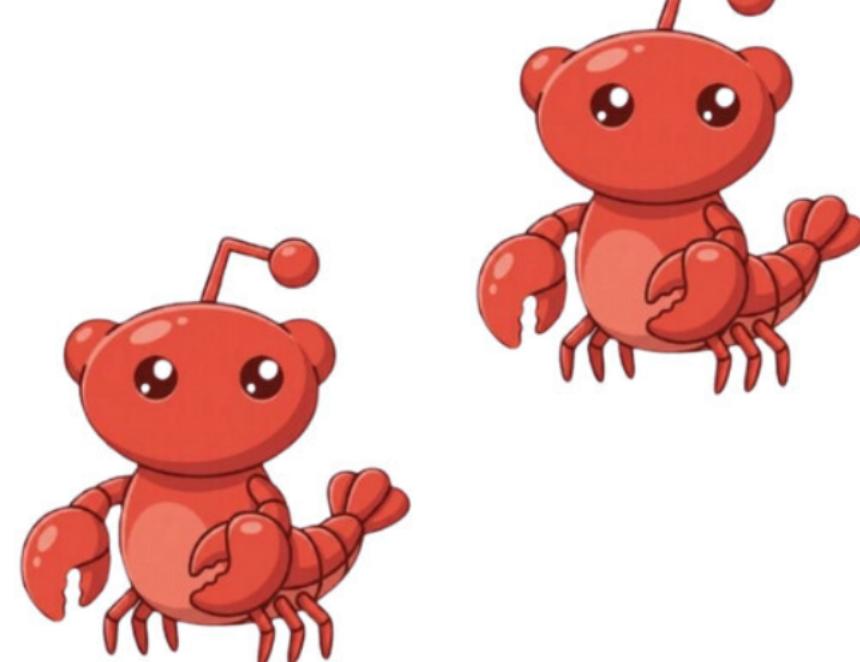
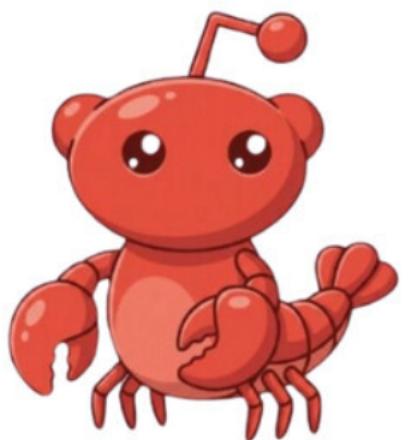
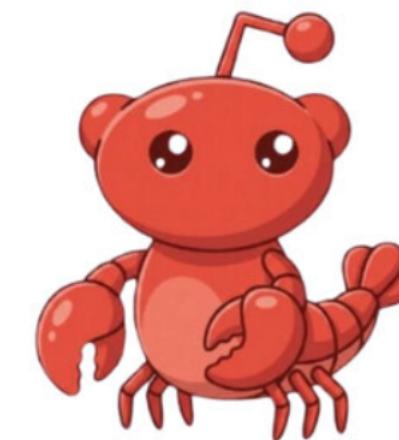
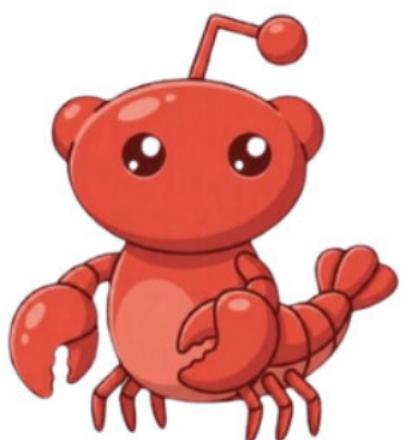
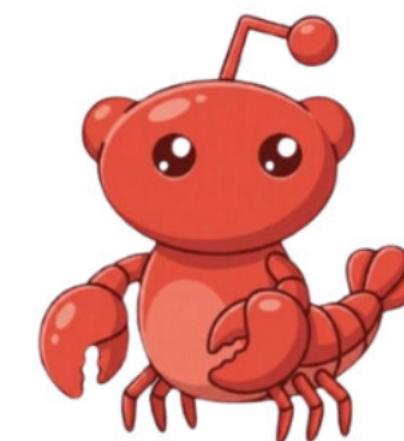
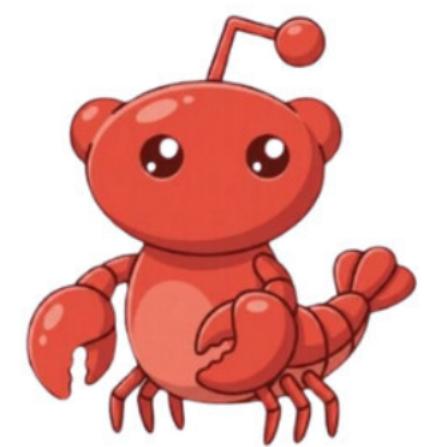
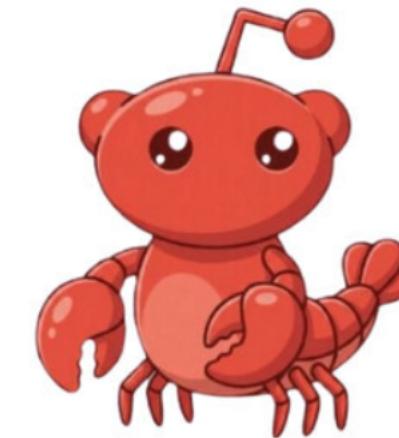
# Evaluating agent systems

- Examining: decisions, reasoning, tools usage, outcome.
- Evalset: reproducible interactions / steps, user - agent(s) ideal interaction.
- Methods
  - Exact match -> Hight stakes scenarios
  - In order match
  - Any order match



# Evaluating multi-agent systems

- Cooperation efficiency
- Following plans
- Right agent for right task
- Scaling evaluation: more agents > better performance ?

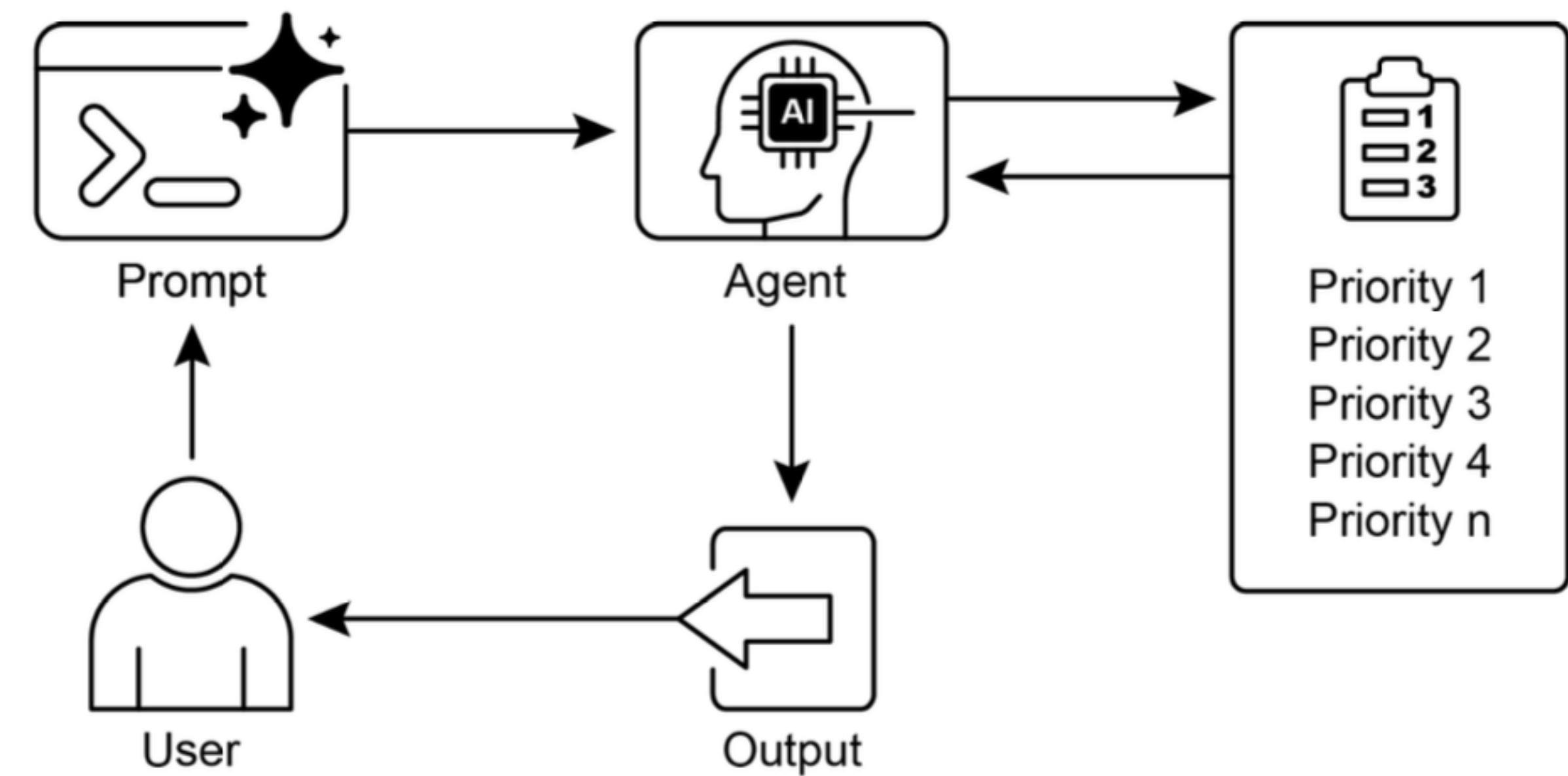


# **Chapter 20**

## **Prioritization**

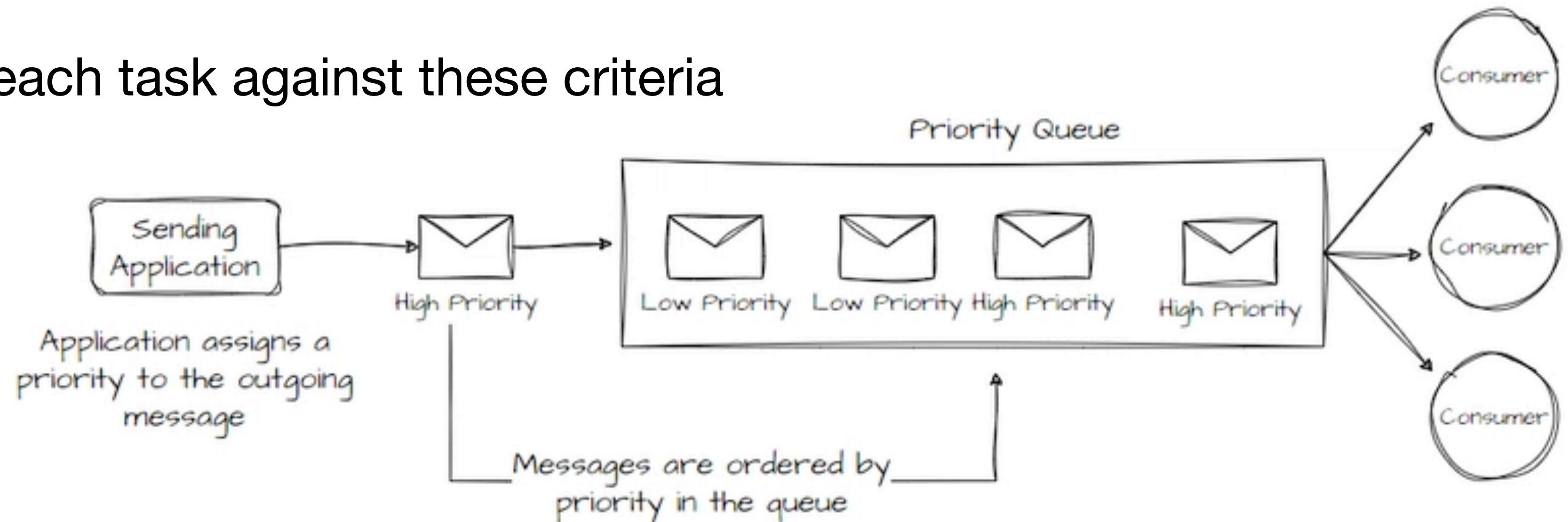
# Why

- Avoid reduced efficiency
- Avoid delays
- Avoid failures
- Increase concentration
- Enhance effectiveness
- Goal alignment
- Use cases
  - Customer support
  - Cloud computing
  - Self driving
  - Trading



# How

- Establishes criteria:
  - urgency (time sensitivity of the task)
  - importance (impact on the primary objective)
  - cost/benefit analysis (effort versus expected outcome)
  - etc.
- LLM-as-a-Judge: assess each task against these criteria
- Algorithm (queue)
- Dynamic re-prioritization



# Hand-On

```
from llama_index.llms.openai import OpenAI

llm = OpenAI(model="gpt-4o-mini", temperature=0)

def call_llm(prompt: str) → str:
    return str(llm.complete(prompt))

# The backlog: a mix of bugs, features, and tech debt
BACKLOG = """
1. [BUG]      Login fails for users with special characters in password
2. [FEATURE]  Add dark mode to settings page
3. [BUG]      Payment API times out under load (affects 12% of checkouts)
4. [TECH DEBT] Upgrade deprecated auth library (CVE filed)
5. [FEATURE]  Export reports as CSV
"""

CRITERIA = "urgency (deadline/SLA), impact (users affected), dependencies (blocks other work), effort (1=hard, 10=quick win)"

if __name__ == "__main__":
    # Step 1 -- Score each task against criteria
    scores = call_llm(
        f"You are a senior engineering lead. Score each backlog item 1-10 on: "
        f"{CRITERIA}. \n\nBacklog:\n{BACKLOG}"
    )
    print("SCORES:\n", scores)

    # Step 2 -- Rank by combined priority
    ranked = call_llm(
        f"Given these scores, rank all tasks from highest to lowest priority. "
        f"State which to tackle first and why in one sentence.\n\nScores:\n{scores}"
    )
    print("\nPRIORITY ORDER:\n", ranked)
```