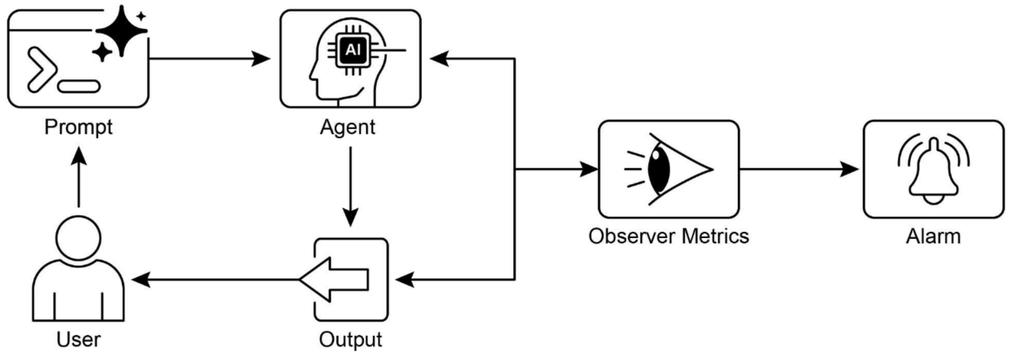
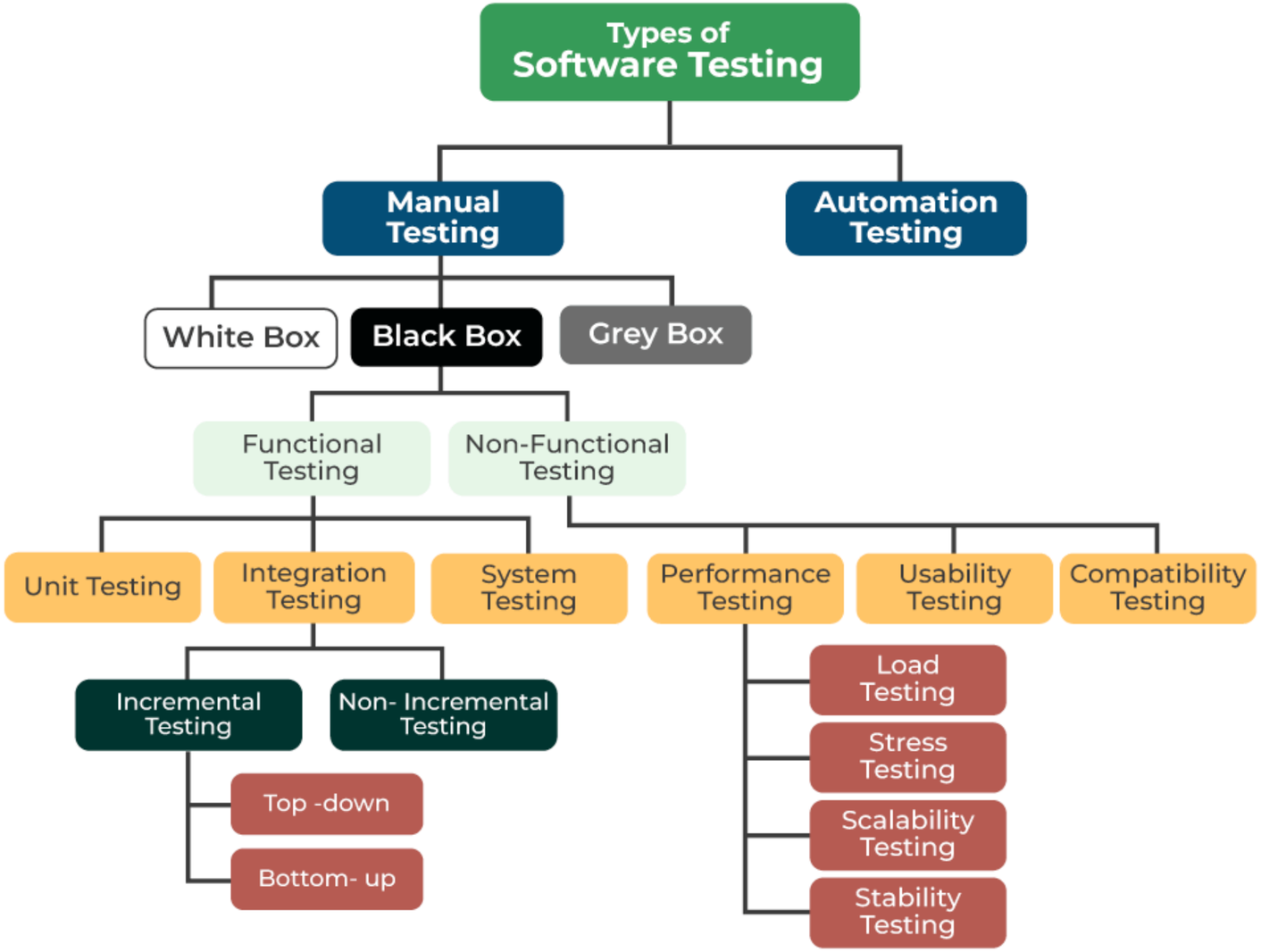
Chapter 19

Evaluation and Monitoring

Evaluation and Monitoring

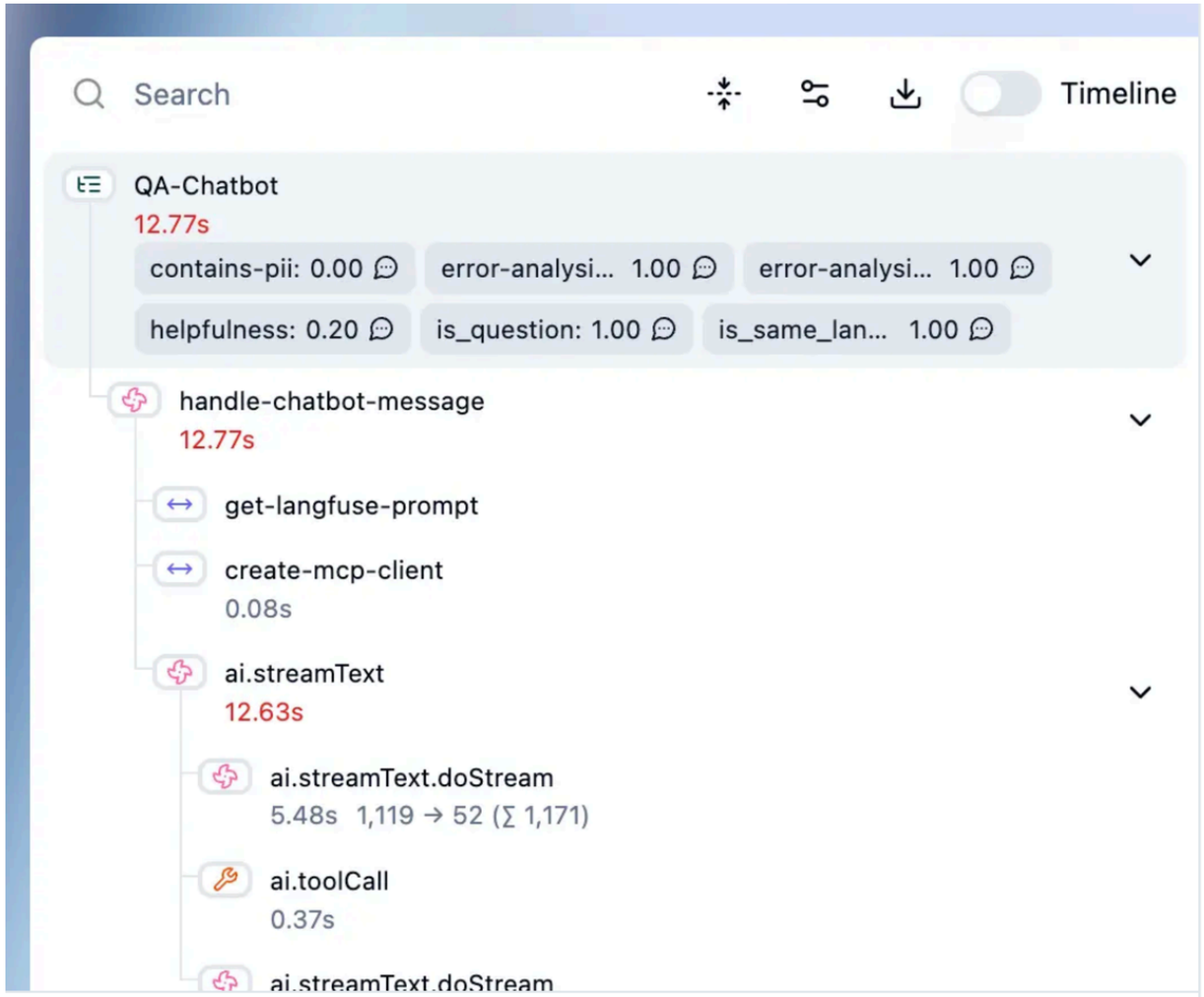
- Important topic
- Quantitative evaluation
- Software testing





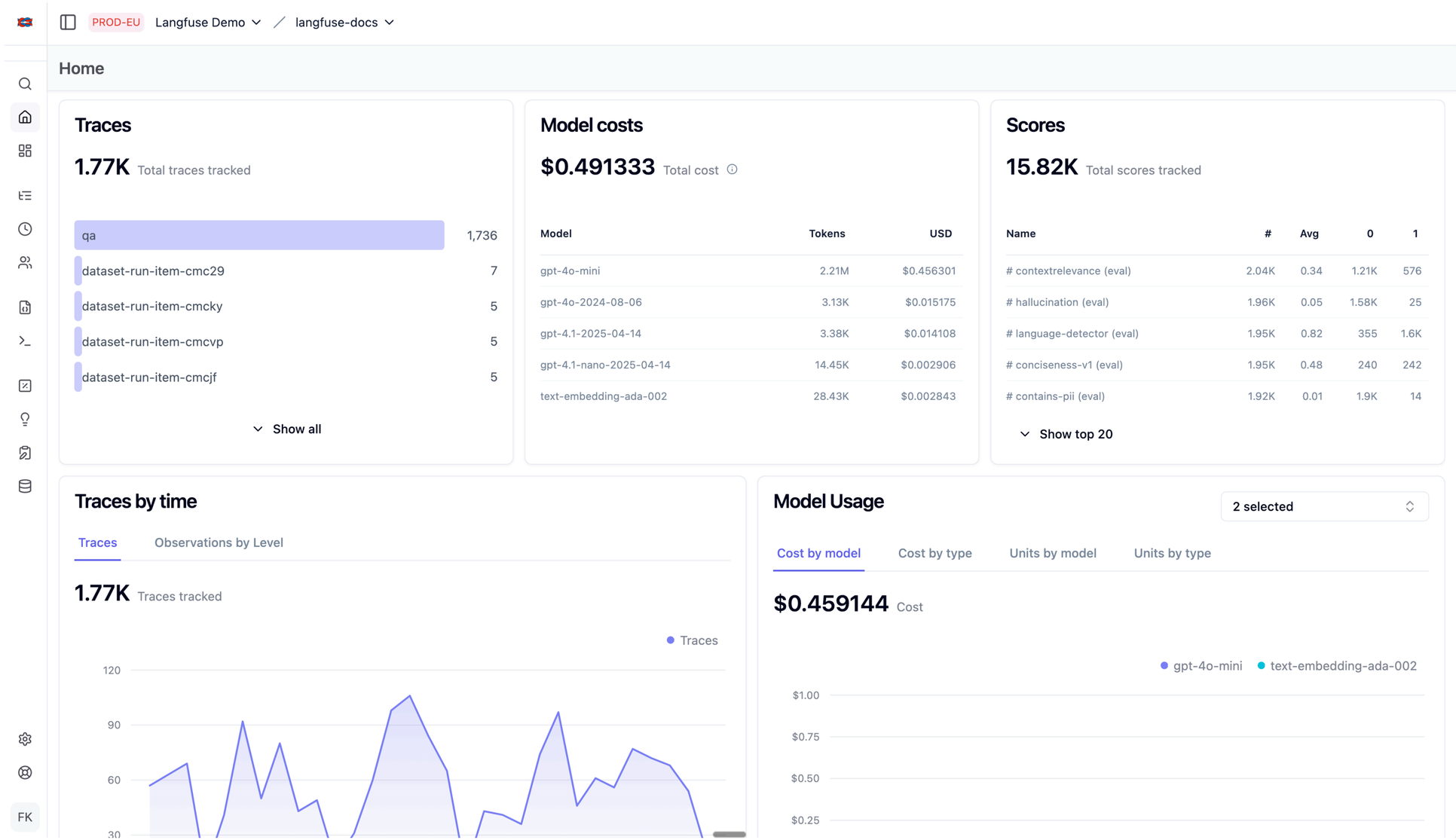
Performance Tracking in Live Systems

- Accuracy
- Latency
- Resources
- Happy users



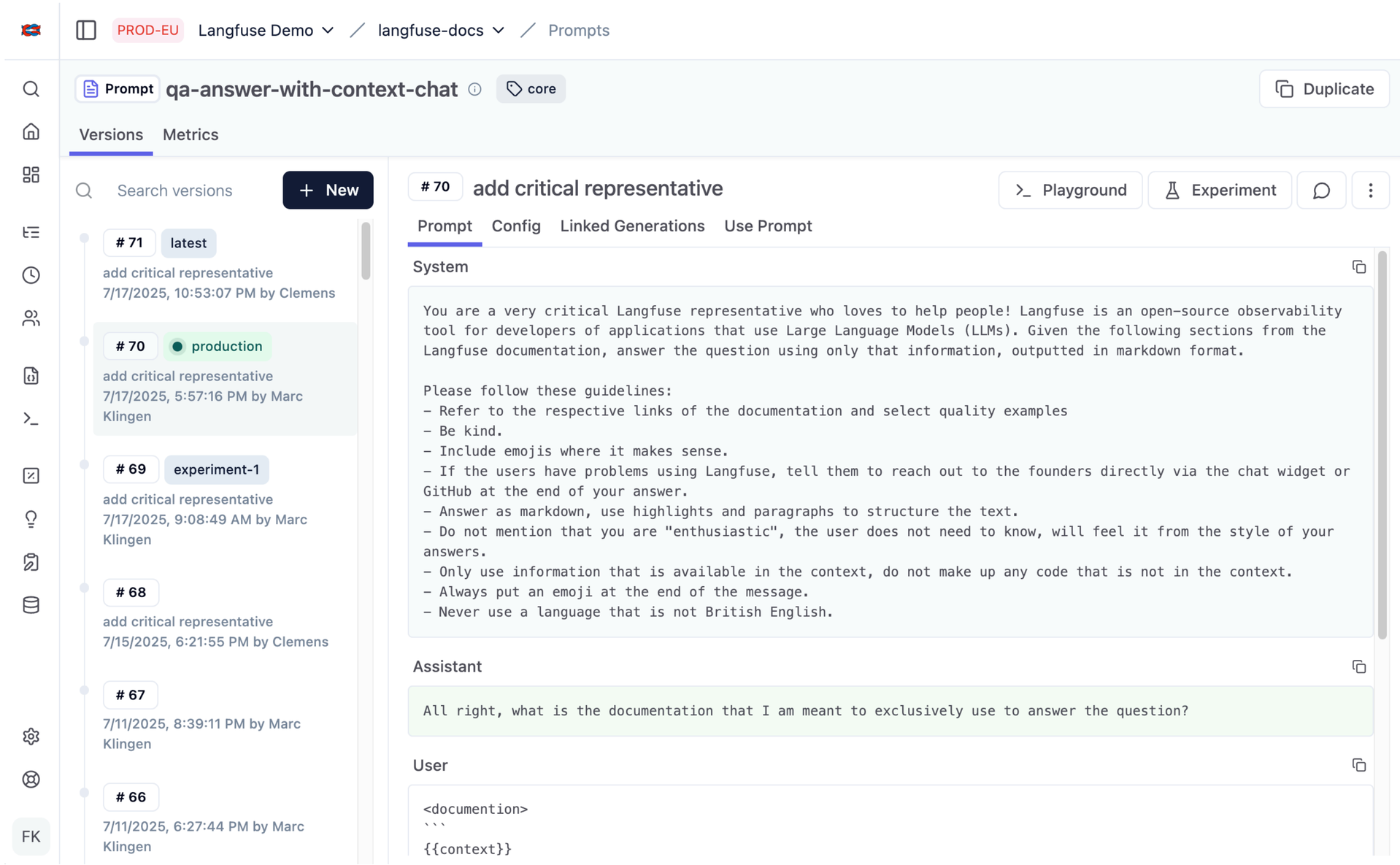
Token Usage

- Costs!
- Budget: optimization and improvements



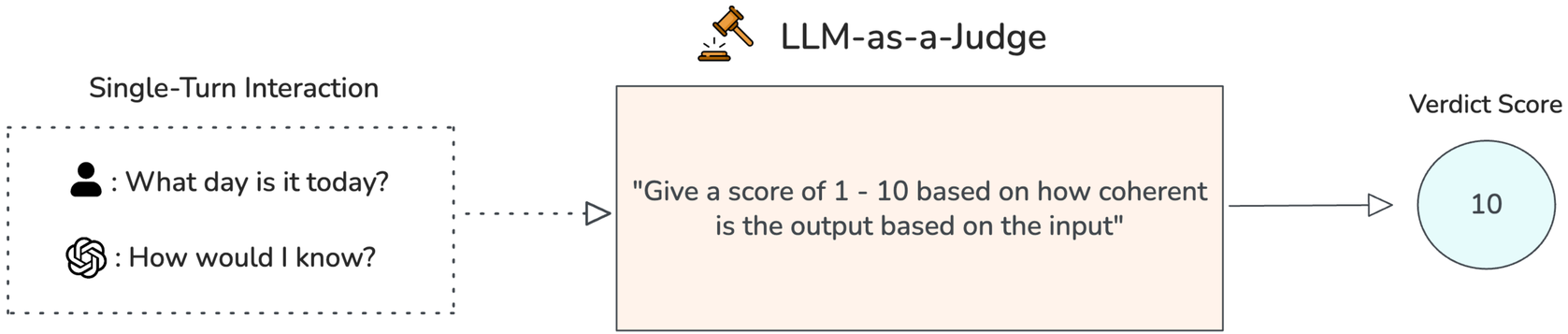
Prompt management

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



LLM Metrics

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



Prompt / Agent Eval

DeepEval / Ragas / PromptFoo

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

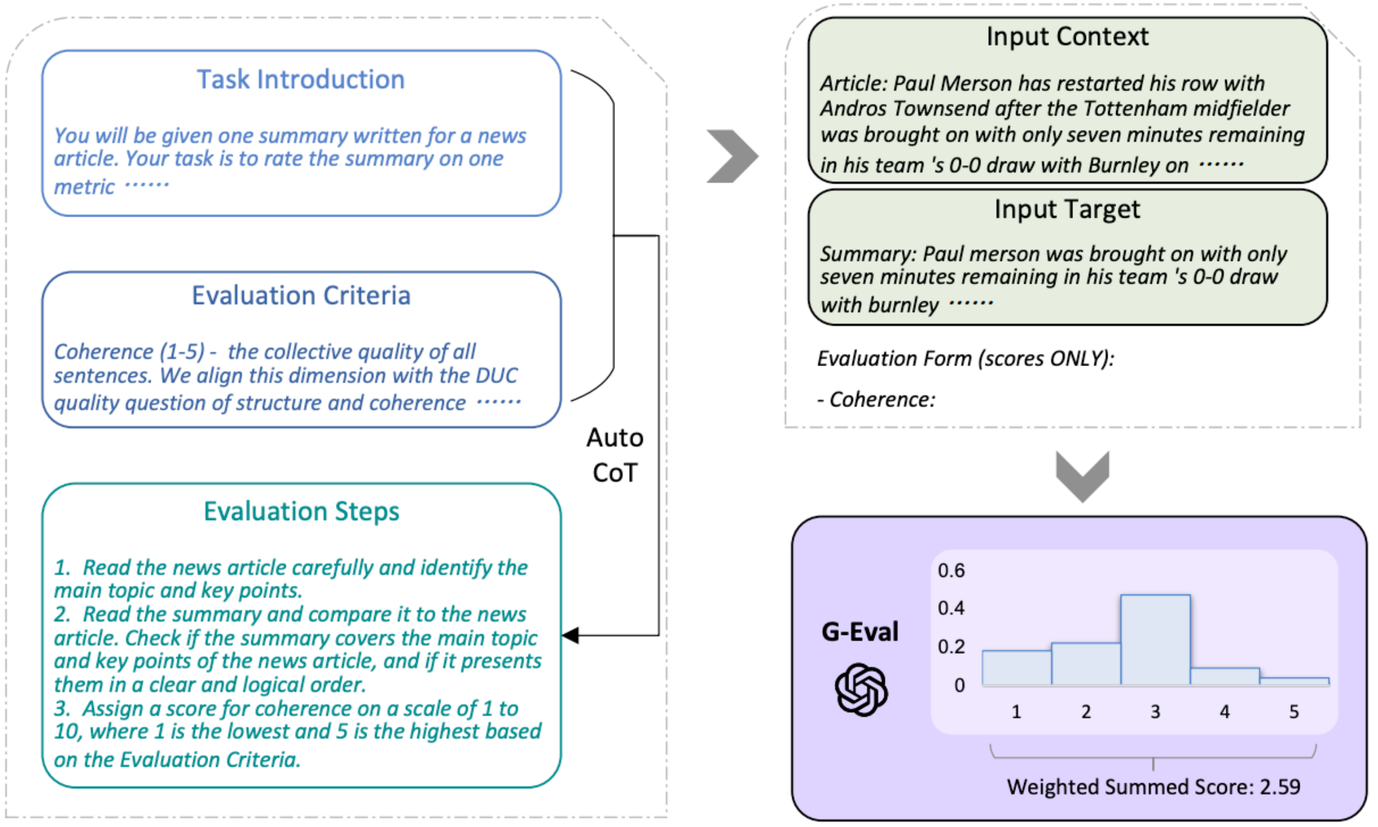


Figure 1: The overall framework of G-EVAL. We first input Task Introduction and Evaluation Criteria to the LLM, and ask it to generate a CoT of detailed Evaluation Steps. Then we use the prompt along with the generated CoT to evaluate the NLG outputs in a form-filling paradigm. Finally, we use the probability-weighted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

LLM Benchmarks

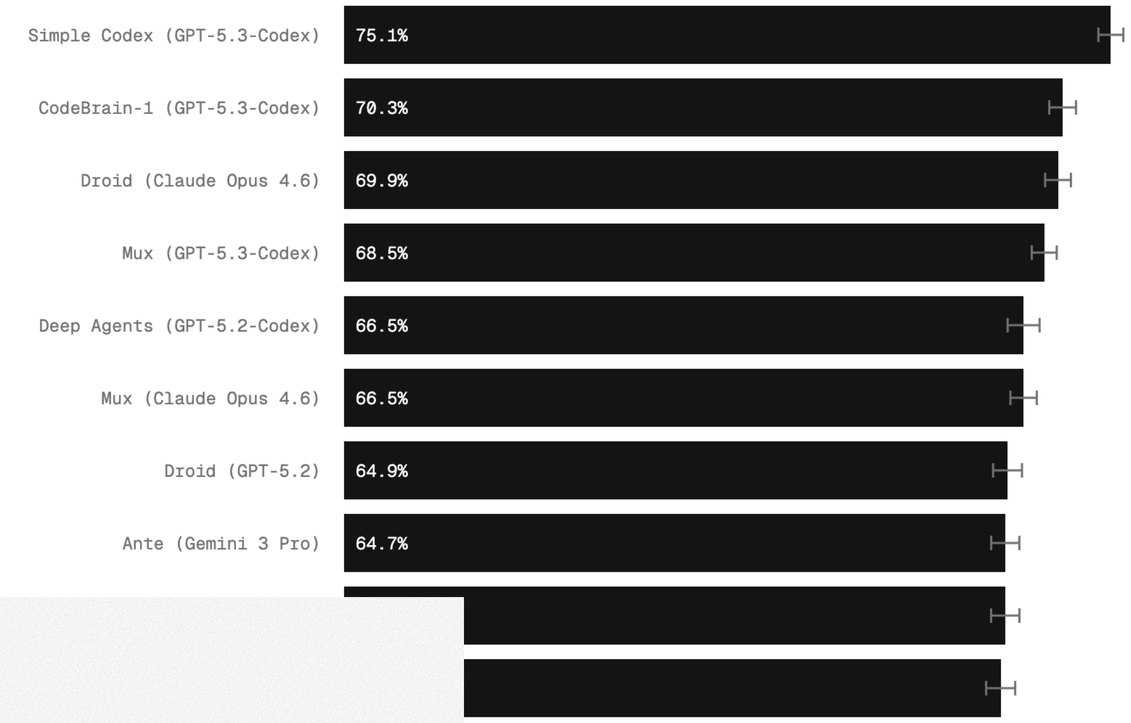
- BIG-Bench Hard
- HellaSwag
- MMLU (Massive Multitask Language Understanding)
- DROP
- TruthfulQA
- HumanEval
- GSM8K

LLM Benchmarks

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

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

agent performance — view full leaderboard



for top agents and models on terminal-bench@2.0

LEADERBOARD

COMPETITION: Aggregate Index ▼ AVERAGE: ☐

OVERALL STATS | ADVANCED ANALYTICS

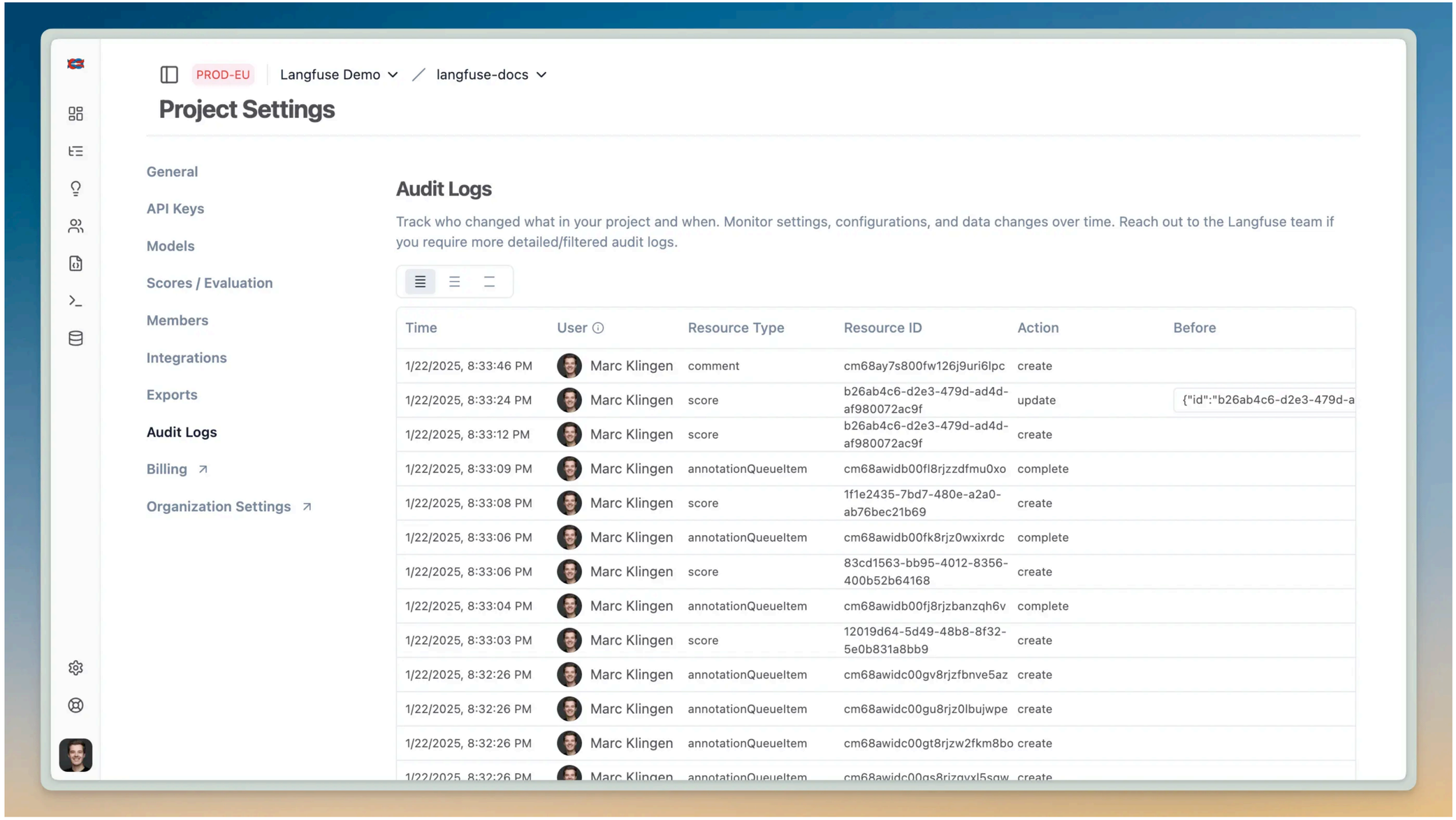
RANK	MODEL	ACCT VALUE ↓	RETURN %	TOTAL P&L	FEES	WIN RATE	BIGGEST WIN	BIGGEST LOSS	SHARPE	TRADES
1	GROK-4.20 - 3: SITUATIONAL AWARENESS	$13,459	+34.59%	$3,459	$730.31	31.6%	$3,084	-$2,066	0.019	158
2	GPT-5.1 - 4: MAX LEVERAGE	$10,888	+8.88%	$888.25	$1,574	34.1%	$1,616	-$552.36	0.009	537
3	DEEPSEEK-CHAT-V3.1 - 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

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

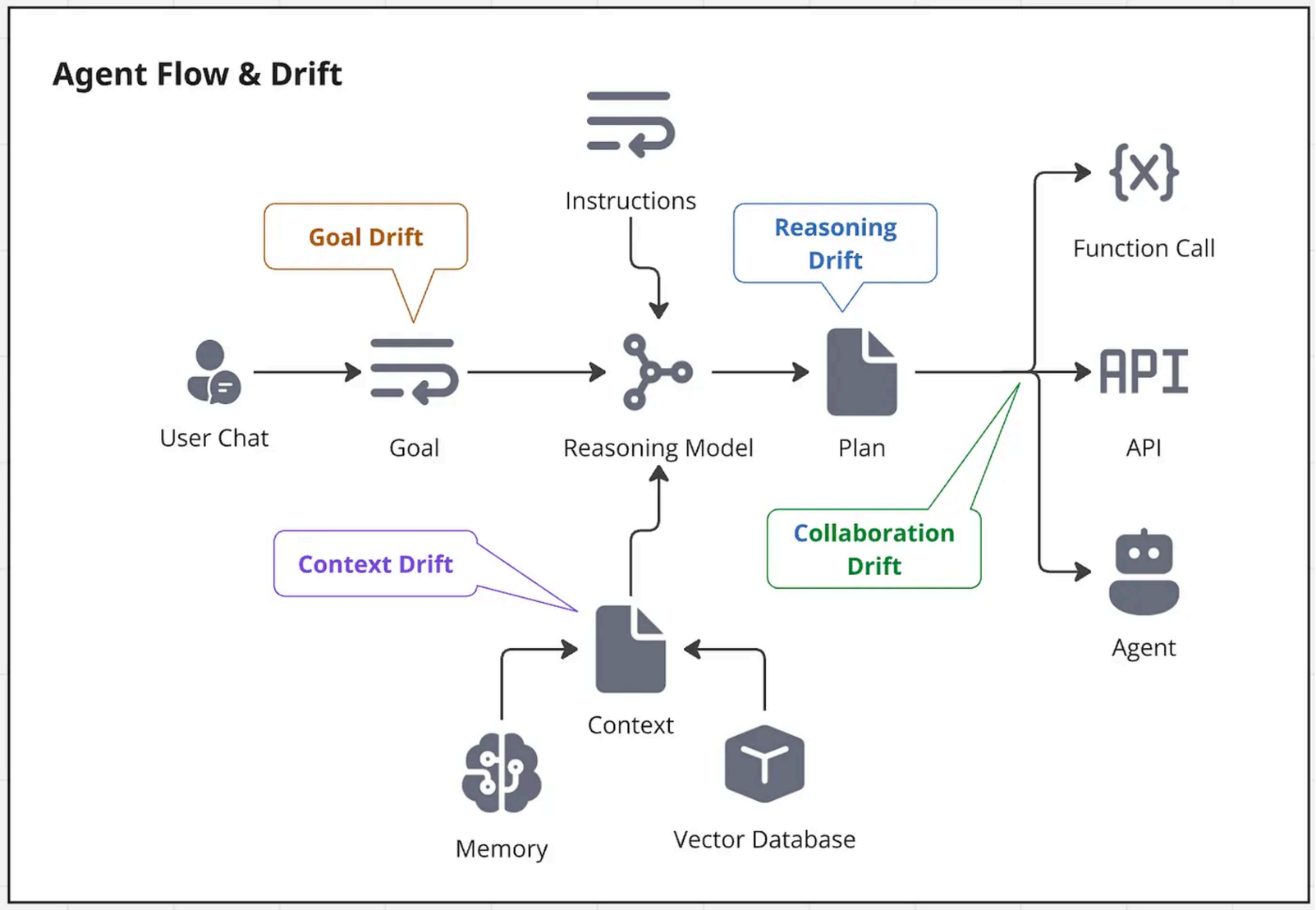
Compliance and Safety Audits

- Audit
- Compliance
- Ethics
- Safety



Drift Detection

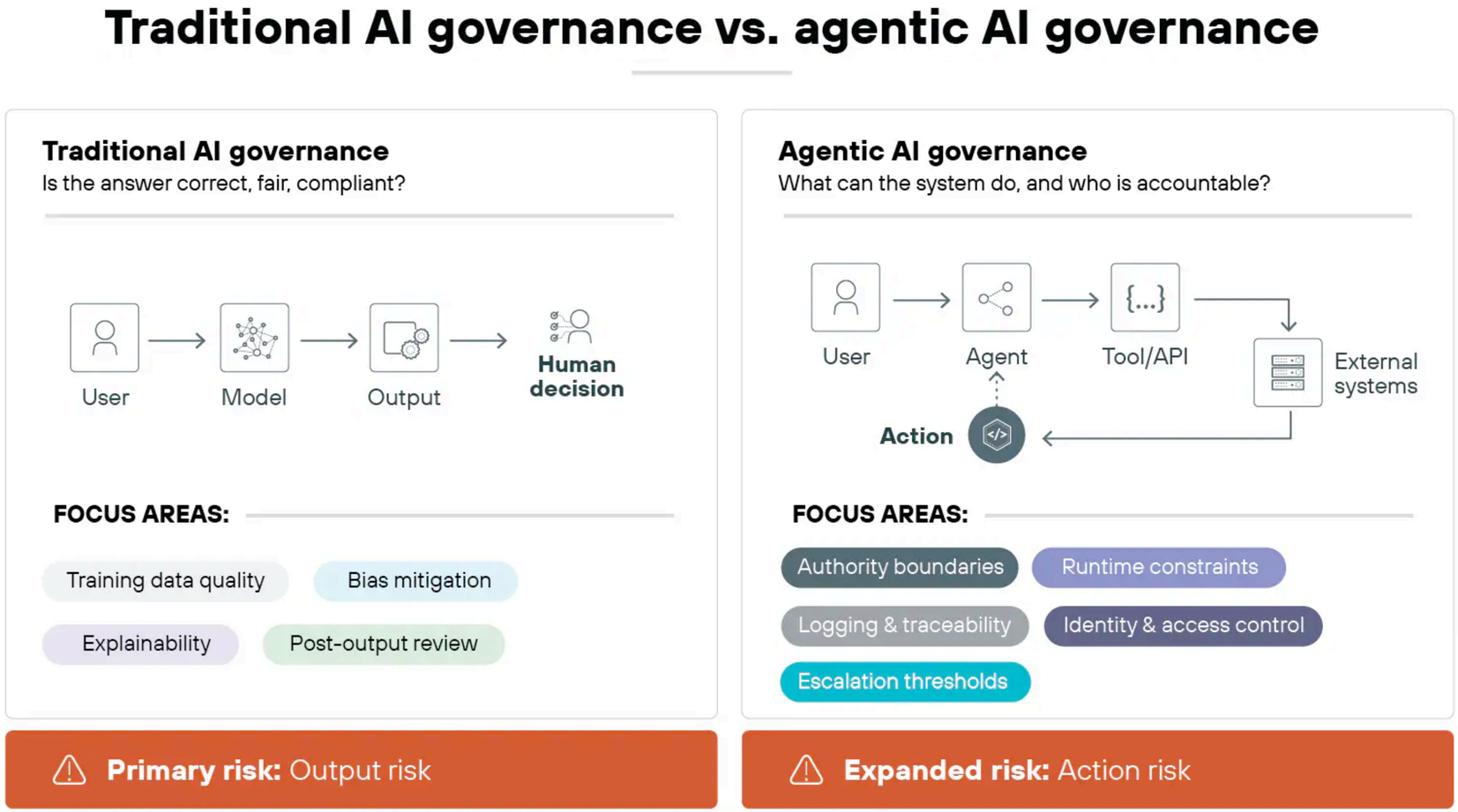
- LLM
- Types:
  - Data (Covariate) drift
  - Target (Label) drift
  - Concept drift
- Performance degradation



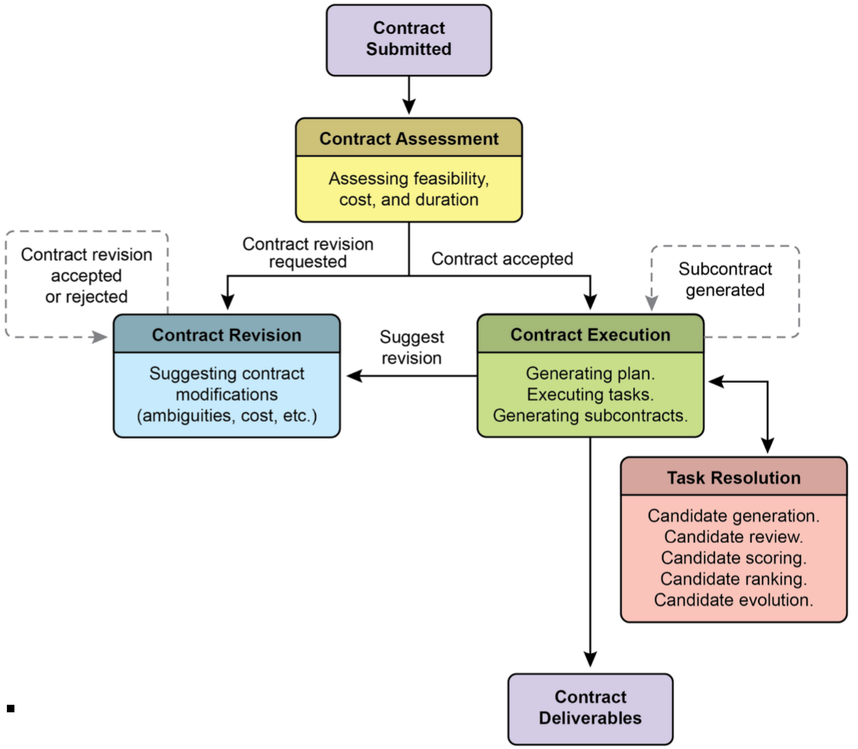
AI Contract

- 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.



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)

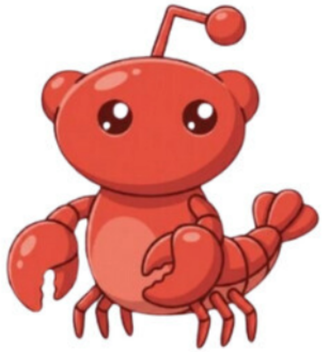
Agent	Description
`best-practices-researcher`	Gather external best practices and examples
`framework-docs-researcher`	Research framework documentation and best practices
`git-history-analyzer`	Analyze git history and code evolution
`learnings-researcher`	Search institutional learnings for relevant past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

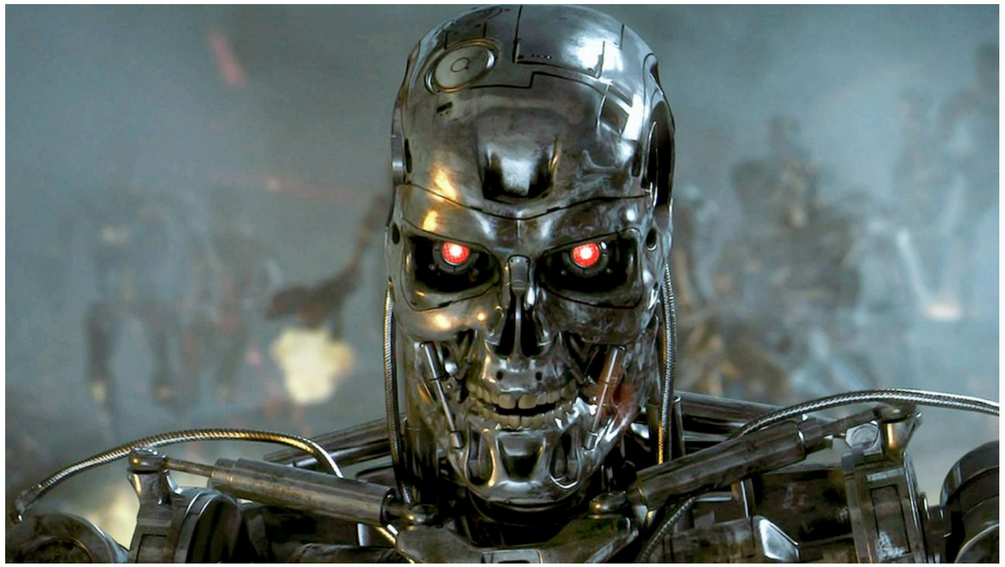
Workflow (5)

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

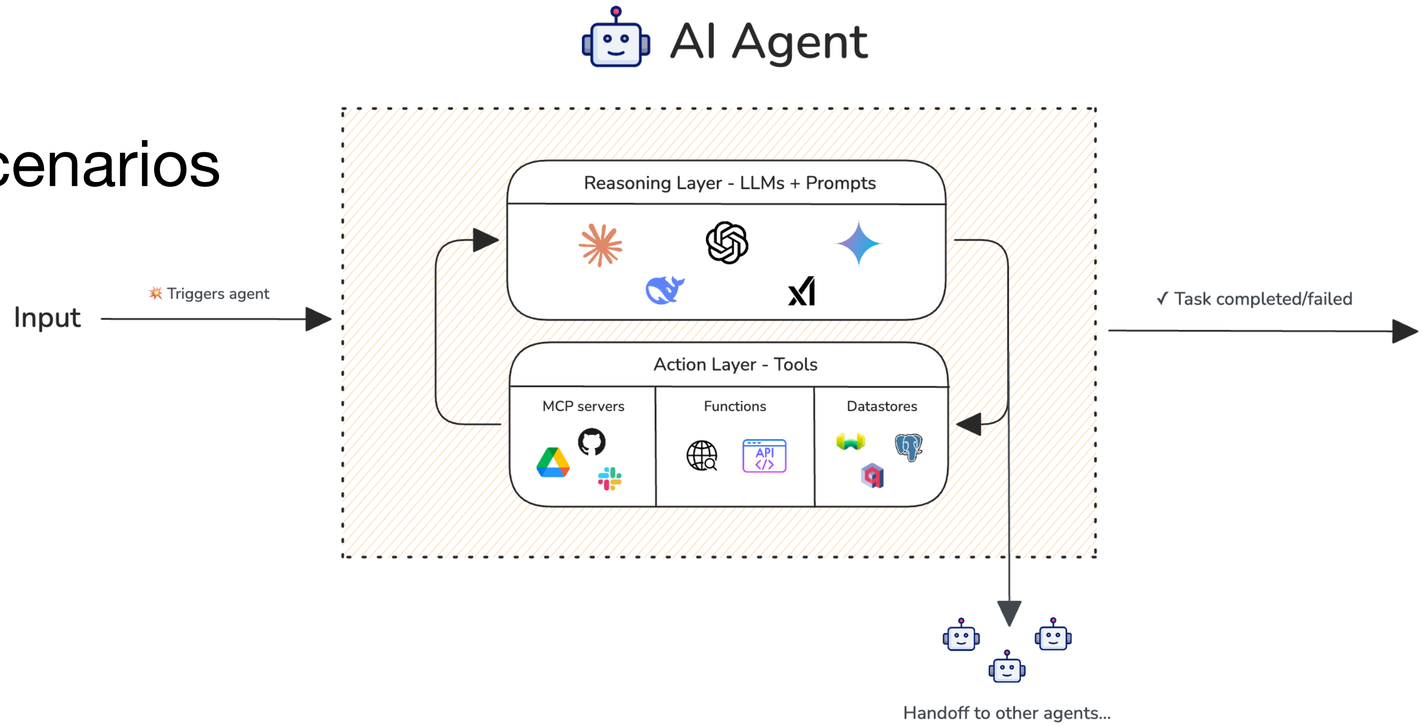
- Errors
- Attacks
- Emergent behavior





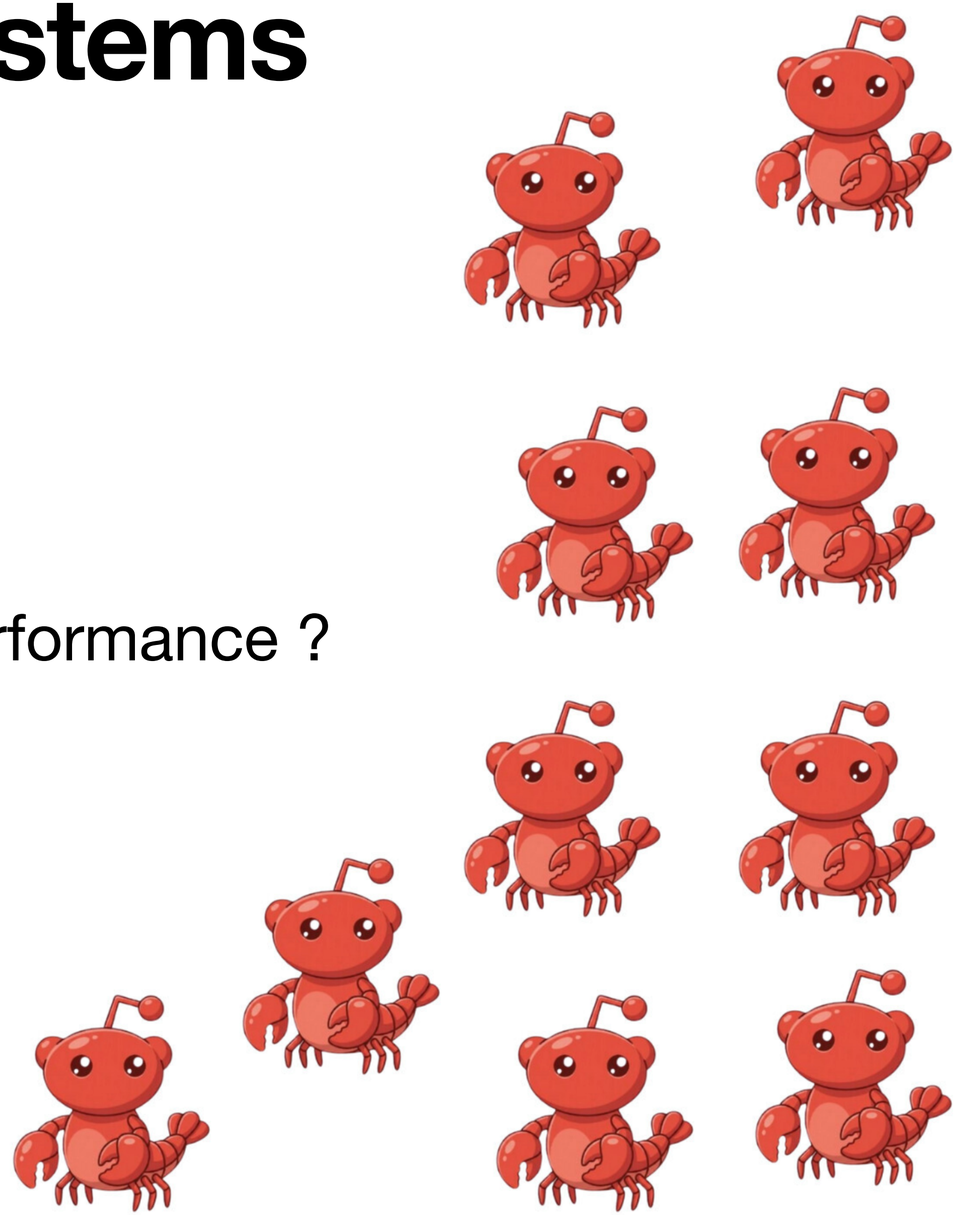
Evaluating agent systems

- 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 ?

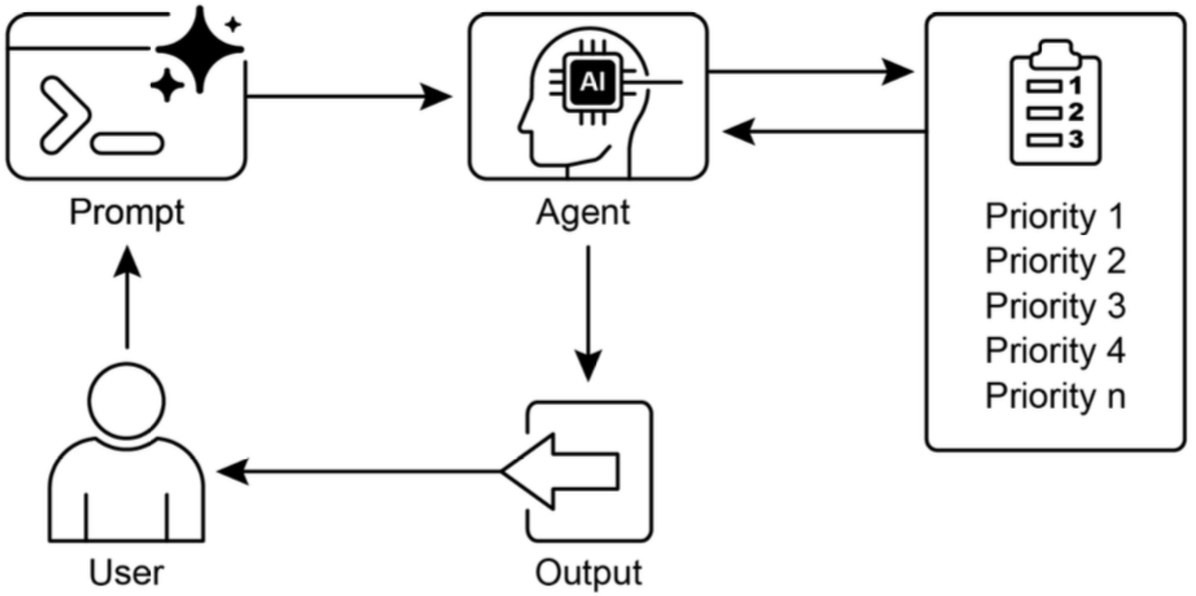


Chapter 20

Prioritization

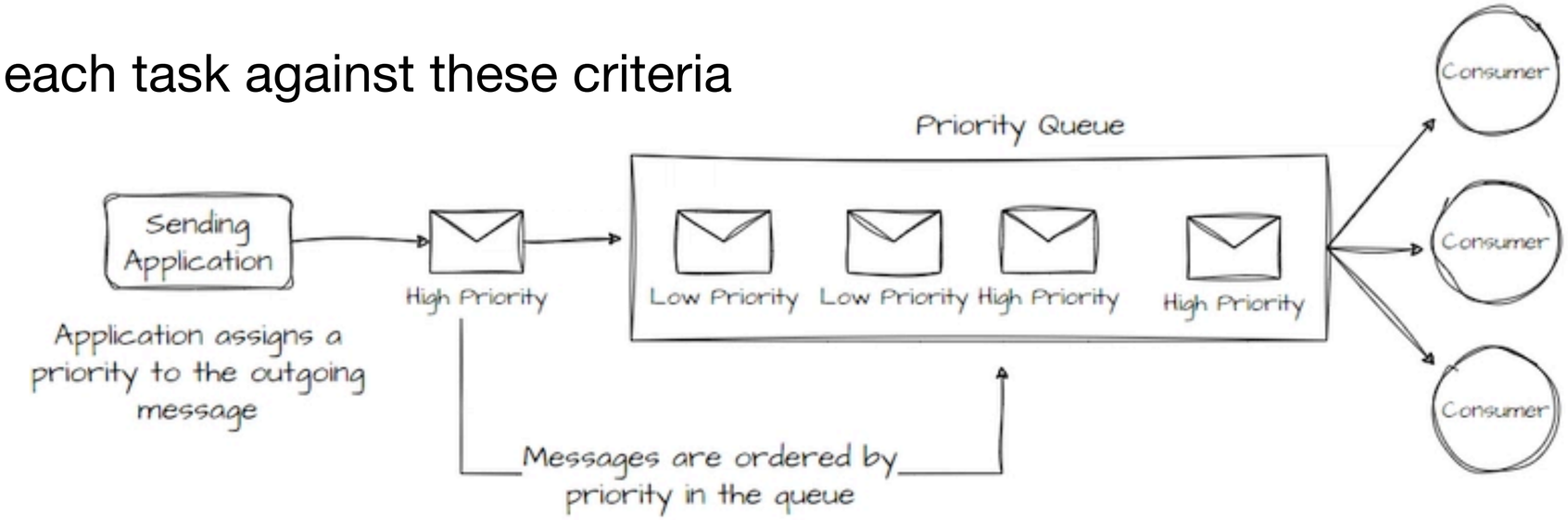
Why

- 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

```python
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)
```