Galileo

The AI Agent Reliability Playbook

Shipping reliable, production-ready AI agents at scale

Anastasiia Biloshytska
Lead DevOps Eng, Galileo

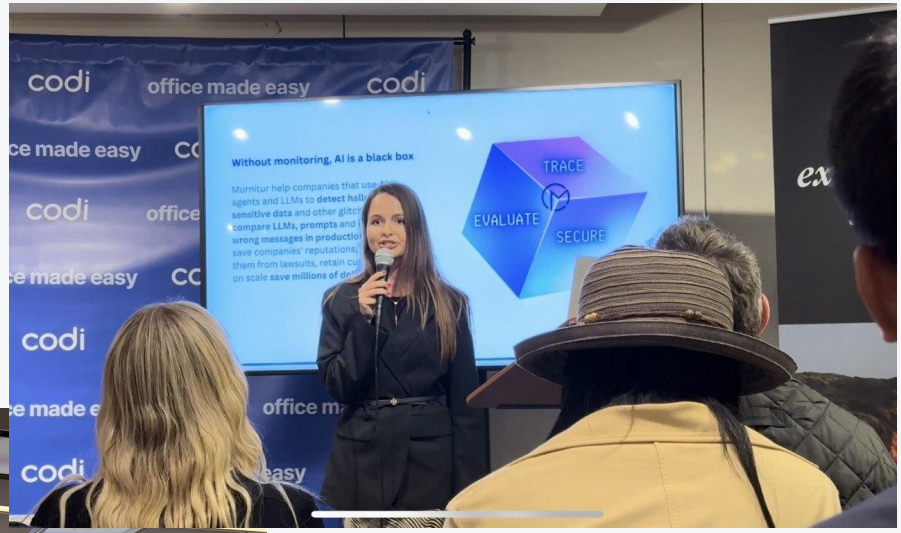
Confidential | © 2025 Galileo Technologies, Inc.

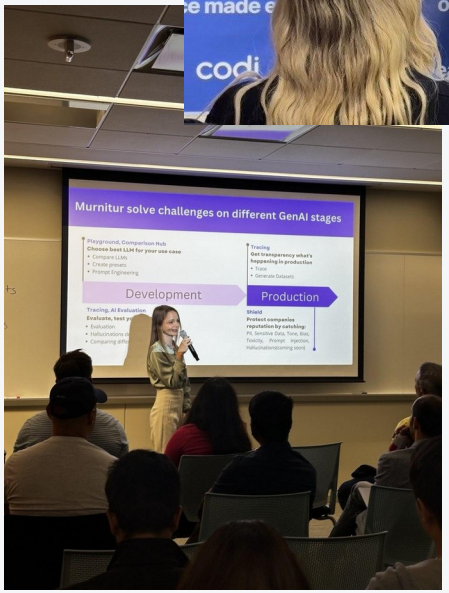
Galileo

Is AI safe?

Murnitur AI


Invest in TAM
Future Available
Market
TRACE
EVALUATE
ANASTASIIA
BILOSHYTSKA
Sponsor


codi
office made easy
Without monitoring, AI is a black box
TRACE
EVALUATE
SECURE


Murnitur solve challenges on different GenAI stages
Development
Production


AI OBSERVABILITY
PLATFORM

Galileo

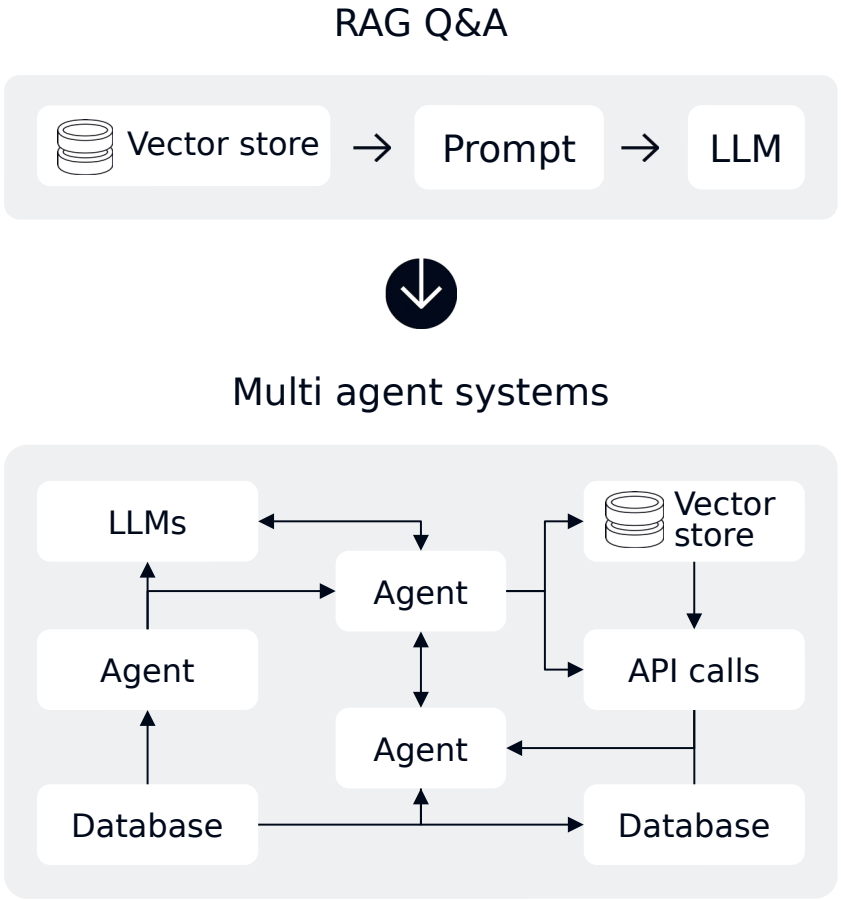
Unified Evals, Observability & Guardrailing for AI Agents


Galileo

+ ?

Why is this Critical?

We've gone from elementary RAG apps to multi agent systems... with almost no ability to evaluate and observe.



What is AI Agent?



Components of an agent

- **Reasoning / Brain / Planning**
  Handles reasoning and planning
- **Memory / Context**
  Remember past interactions as context (for better decisioning)
- **Tools**
  Do things in the world using traditional software paradigms

  → Data retrievals (web search, db lookups)
  → Taking an action
  → Orchestration (agentic networks - calling other agents etc)



3 obstacles to reliable agents

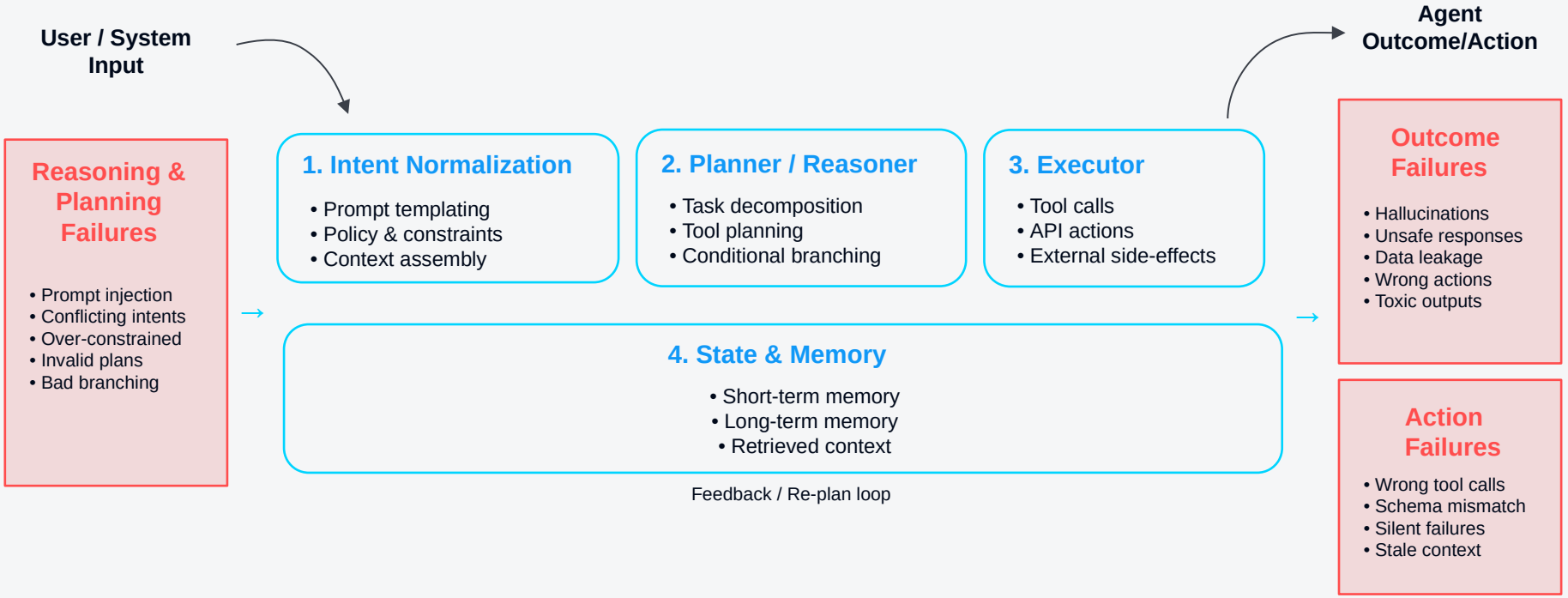
1. Unpredictable interactions (Tool/Memory errors)
2. Over-constrained / Under-constrained Prompts
3. Tool Errors (Tool Selection Issues, Incorrect Tools)

The compounding effect of tiny errors

AGENTIC COMPLEXITY

Agents = Stateful Control Loops

Failures emerge at boundaries between planning, execution, tools, and state.

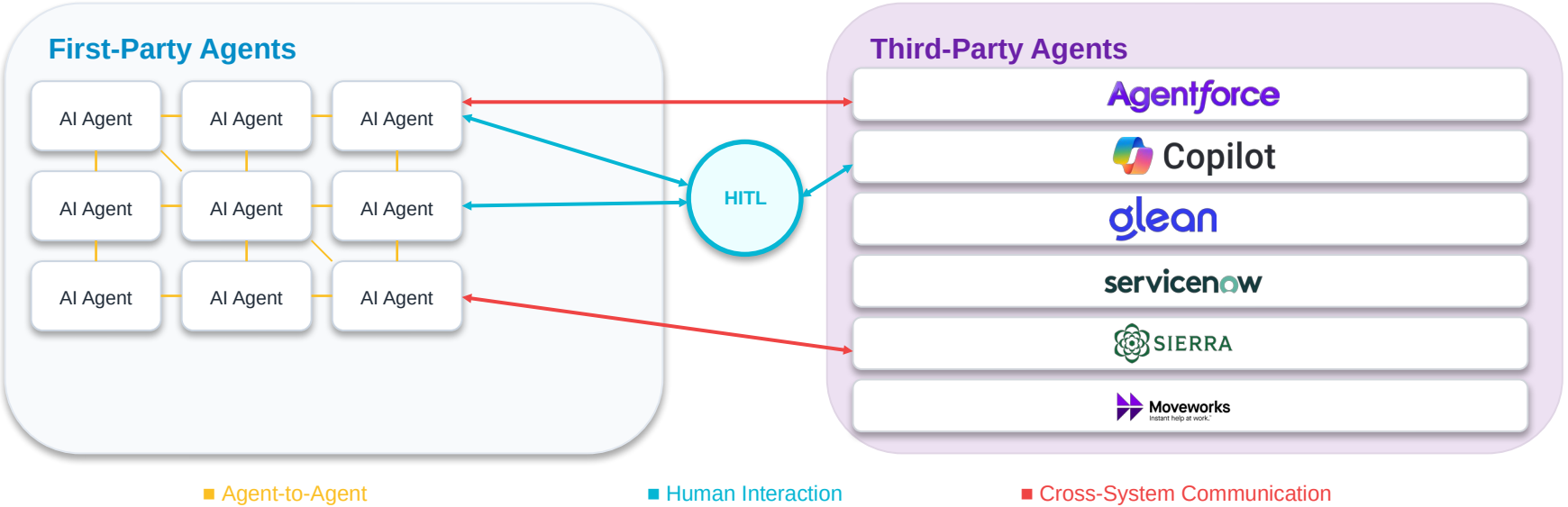


In AI Agents, failures are imminent — without full-loop observability, teams cannot prevent them.

MULTI-AGENT COMPLEXITY

Trust & Control for Heterogeneous Multi-Agent Systems

This interconnected complexity requires unified observability and control

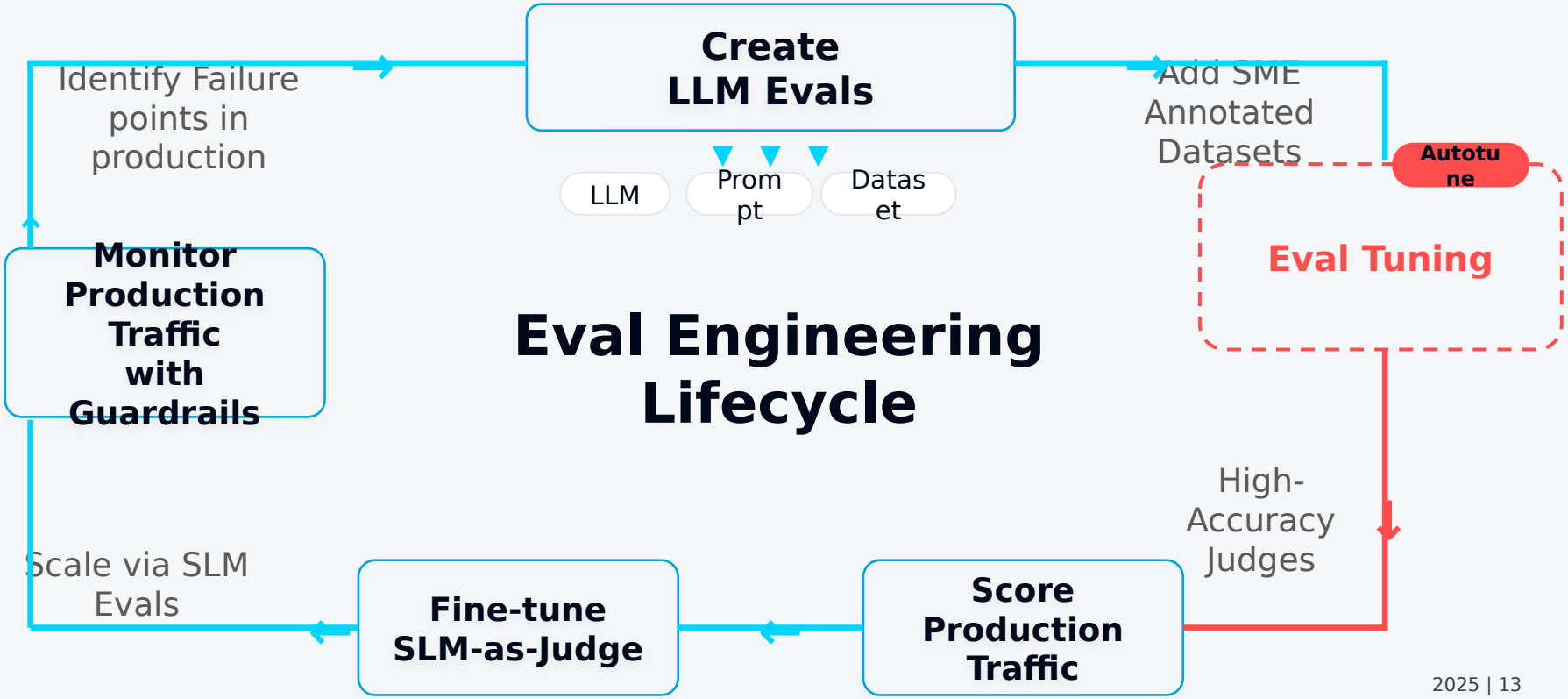


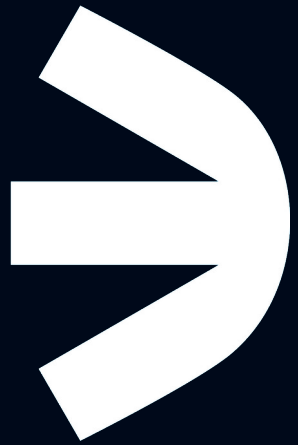
Unified Observability & Guardrails Across All Agent Interactions



How can we build high quality agents?

Eval Engineering





Playbook for Agent Reliability

When to start?

Build Evals from day 1

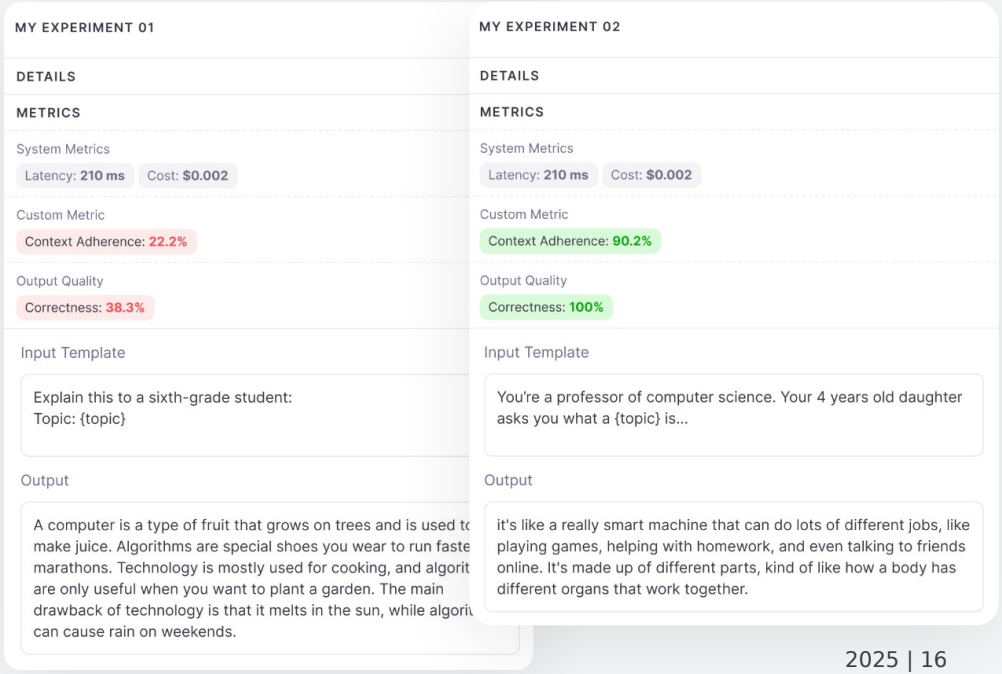


Play 2

Set up Experiments for Simulations



Testing A vs B



Experiments

Essential for identifying, root causing and isolating issues to specific data segments, LLMs, or RAG and agent parameters

Setting up your infrastructure to allow for fast experimentation

Play 3

Optimize your prompts

1

Treat Prompts like Code.

Minor tweaks cause agents to behave erratically.
Treat prompts like code.

2

Bake in Version control

Track changes across prompt iterations.
Maintain lineage.

3

A/B Test & Compare Prompts

Treat prompt evolution like ML experimentation: Run A/B tests on prompt variants against fixed datasets, or vice versa.

Play 4

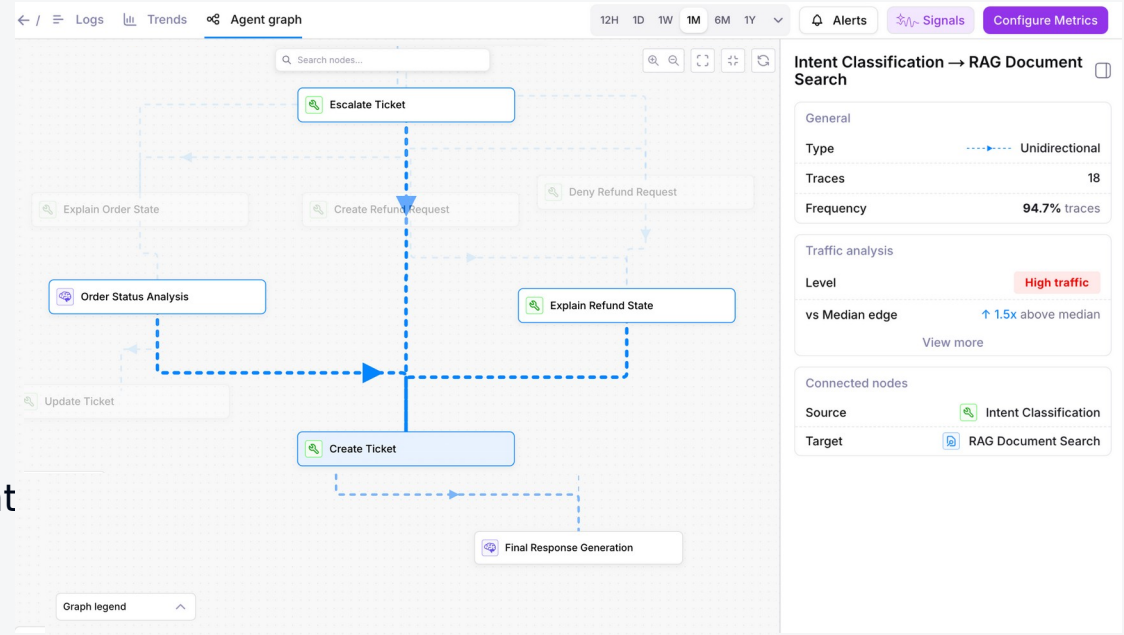
Set up Analytics for Agent Sessions

Agent - Agent

Human - Agent

Agent - Human - Agent

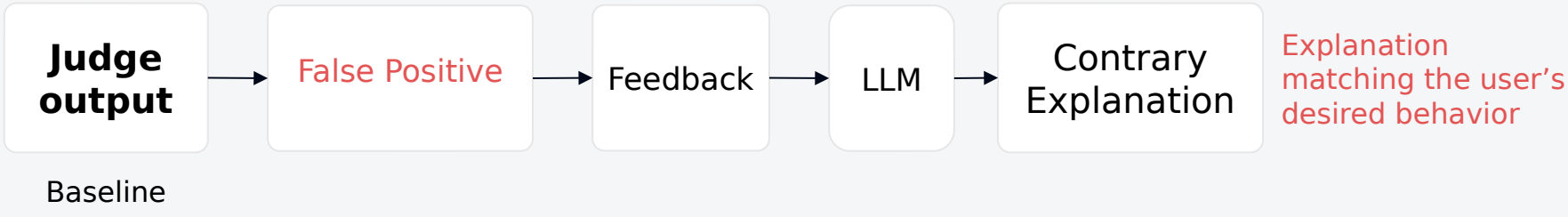
- Traditional tracing is insufficient for multi-agent workflows
- Session Tracing allows for debugging long running agent interactions
- Signals at 3 levels — Individual Requests, Sessions and Application

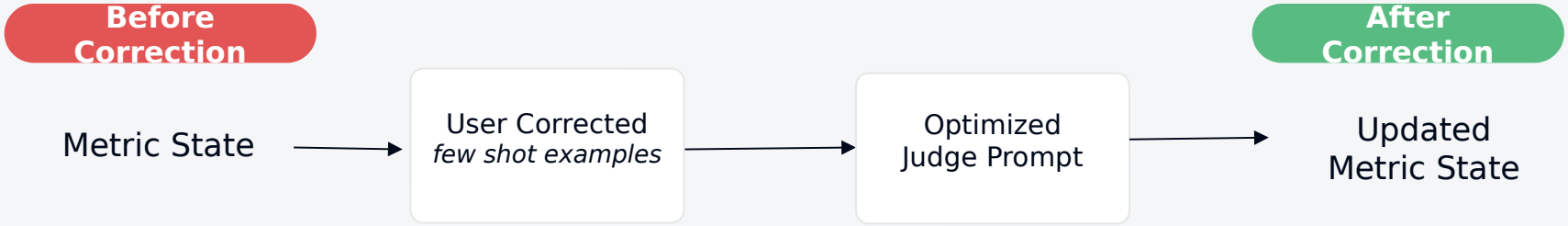


Play 5

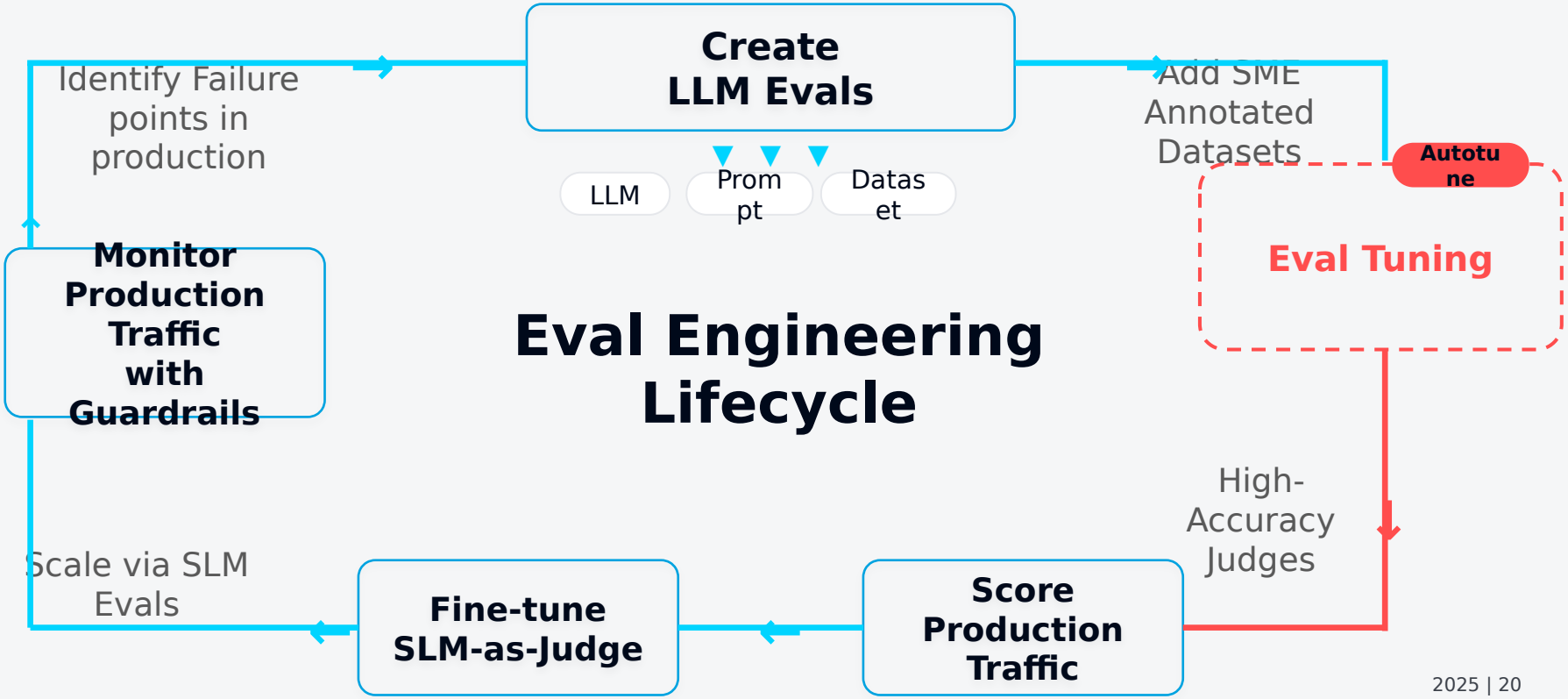
Systematize Human Feedback

Metrics should adapt on continuous basis





Eval Engineering



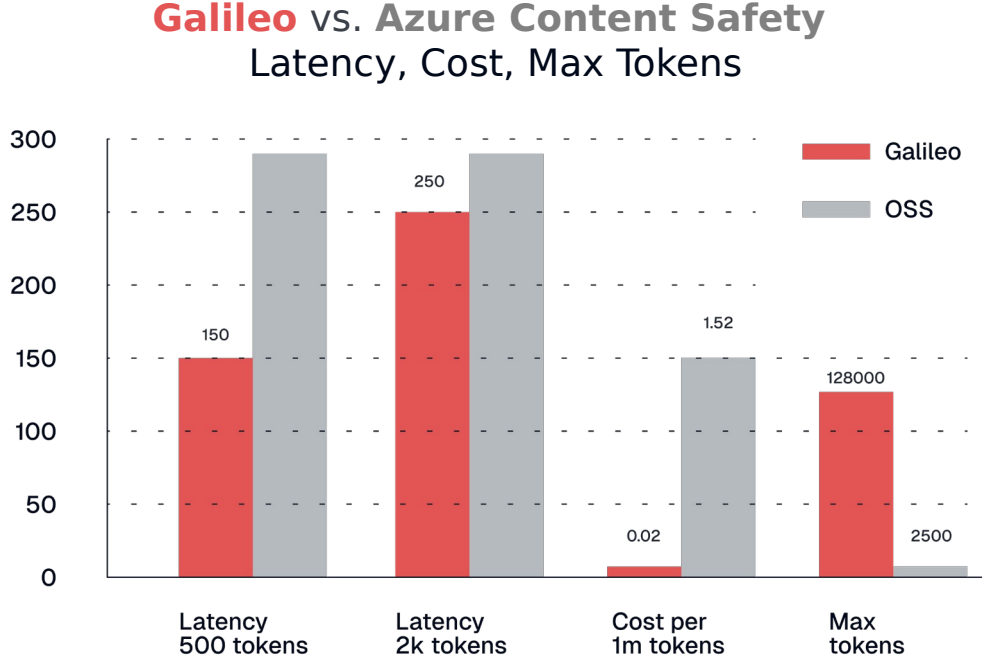
Play 6

Build SLMs for real-time Monitoring & Guardrailing

The Problem?

LLM Judges

do not scale



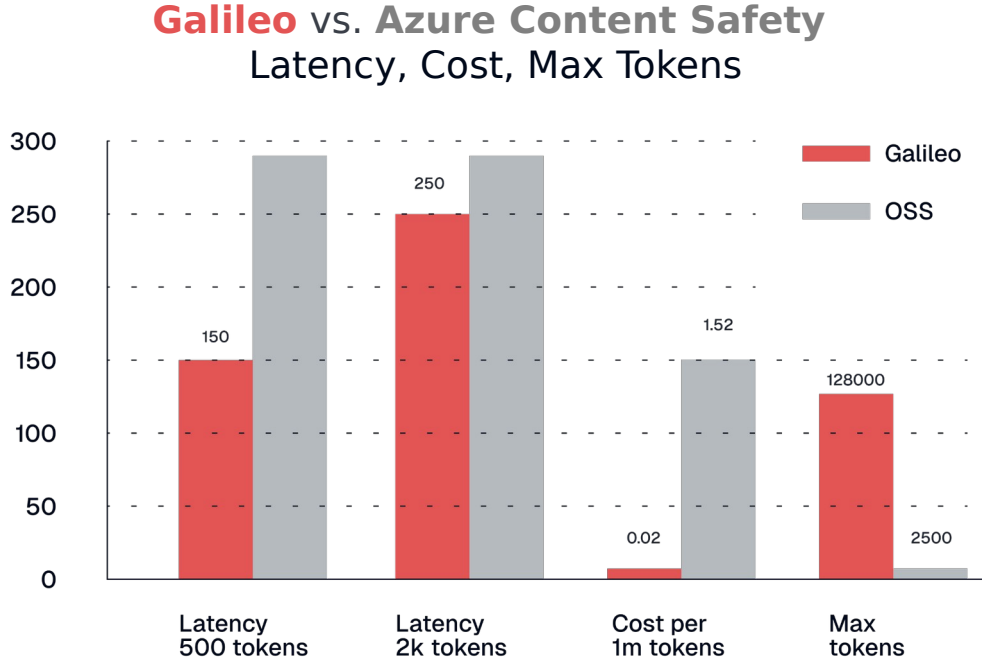
Play 6 cont.

Build SLMs for real-time Monitoring & Guardrailing

What works?

SLMs (1B, 3B, 8B)
LoRA Fine-Tuned

- Transactional Memory + Last level caching, Edge storage for LoRA weights
- Models with shared LoRA weights for maximum utilization & throughput



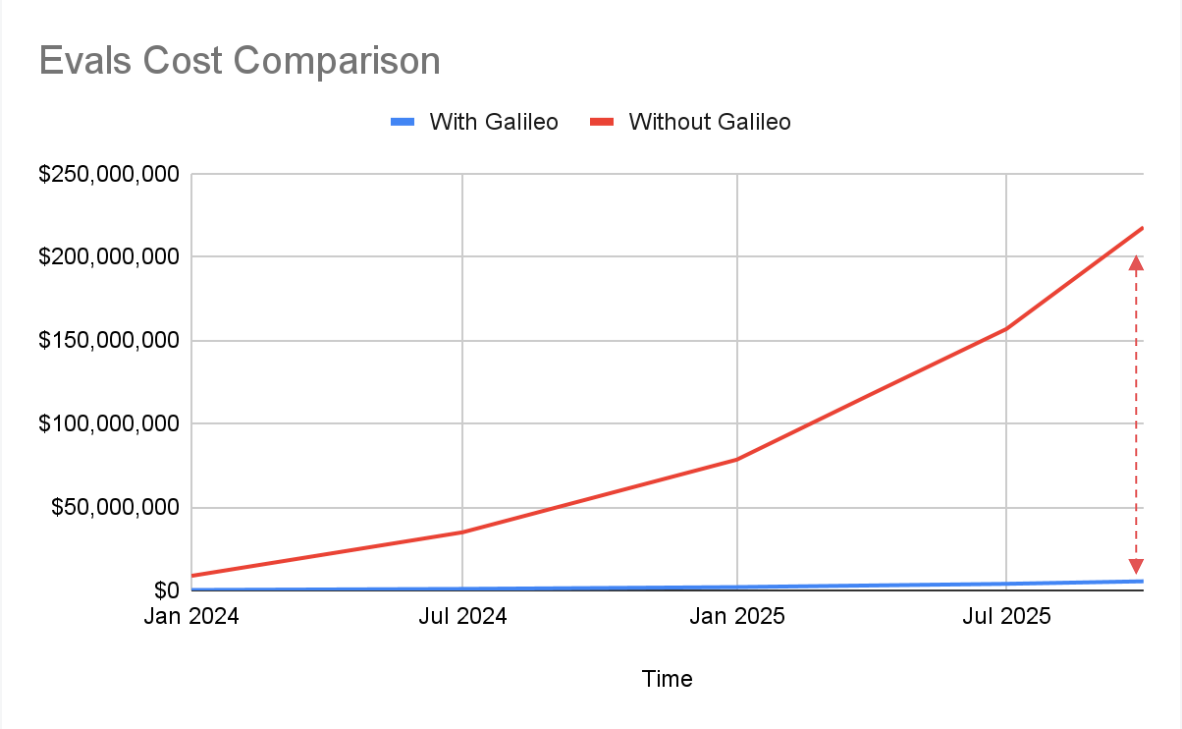
Play 6 cont.

Build SLMs for real-time Monitoring & Guardrailing

What works?

Single Token Probability

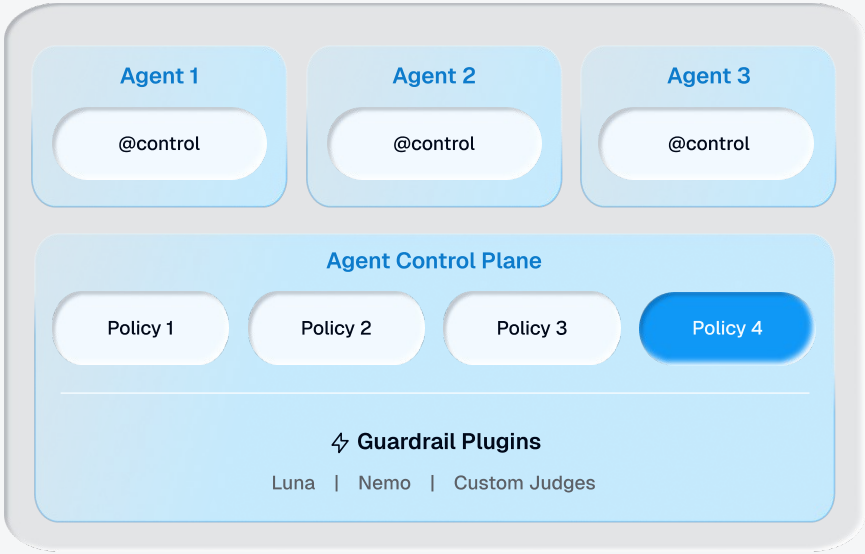
- Compressing outputs into a single token
- Optimized for cost & latency - Runtime evaluation of Agents
- Closed-loop-learning from production data



Play 7

Create Runtime Controls

From Reactive Observability to Proactive, Centralized Controls.



- Standardized Agent Control Policy
- Dynamically updatable
- Access to standard guardrail plugins

Pip install -agent control

```
@control()
def execute_tool_call(tool, args):
    # Policy references the same metric
    return tool.execute(args)
```

Centralized Control Plane

-> Your AC Server acts as the single watchtower across all your agents

Reusable Control Store

→ Build once, reuse across developers

Runtime steering and blocking

-> Build Control Policies with runtime actions baked in

Policy Tracking

-> Track policy enforcement and effect

Galileo



agentcontrol.dev



+



Galileo

Cisco to acquire Galileo

Galileo

Visit galileo.ai

Let’s connect:



Anastasiia Biloshytska

Product Market Analyst | Ex-Founder of Murnitur (Security Platform for LLMs) | Senior...

Confidential | © 2025 Galileo Technologies, Inc.