PagerDuty

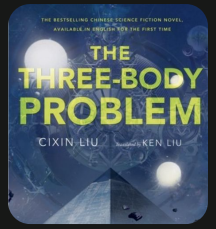
Production AI agents:
The gap between promise and reality

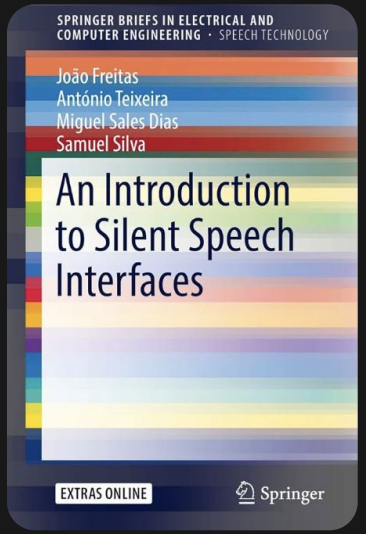
June 2nd 2026

You can build a functional, interactive software prototype in minutes to hours…

About me…

DEFINED.AI





さらに、カリフォルニア州が6月に新たな個人情報保護法を可決したほか、連邦議会でも、データ保護法案の動きが出るなど、GAFAのおひざ元の米国でも包囲網が広がる。

「企業があなたのすべての情報を手にすれば、驚くべき予測ができるようになる。いま議論されているのは、企業がどこまでデータを手にすべきかだ」。AIの機械学習向けのデータ提供を手がける、米ディファインド・クラウドのジョアオ・フレイタス最高技術責任者(CTO)は言う。

人工知能（AI）向けの機械学習向けのデータ提供を手がける、米ディファインド・クラウドのジョアオ・フレイタスCTO（左）＝五十嵐大介撮影

ポルトガル人のフレイタスは、首都リスボンに拠点を置く。欧州ではプライバシー保護に敏感な企業が多いといい、大企業によるデータ独占を「危険なゲームだ」と指摘する。「企業からサービスを受ける代わりに、利用者はどこまでデータを企業に提供するかで同意することが重要だ。利用者が満足できれば、ウィンウィンの状況を作ることができる」

PagerDuty


AI Agents
Feb 30,
19:00
Step
47
of 3
DO NOT DEPLOY TO PROD
DO NOT DEPLOY TO PROD

Real data = Challenges

“Teams often discover that what works in demos breaks once agents interact with real users, real data, and real systems”

Outputs are probabilistic — the same input can produce different actions

Small reasoning errors compound across multi-step workflows.

Agents may hallucinate facts or misunderstand tasks.

Long tasks degrade performance over time (“context fatigue”)

Agents depend heavily on data quality

Context windows overflow

Agents forget earlier instructions

Memory becomes corrupted (“context poisoning”)

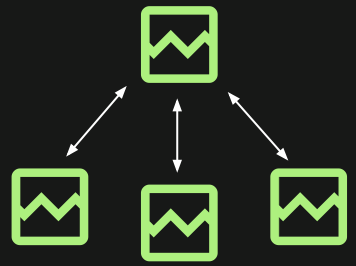
This leads to confident but wrong decisions.

Orchestration Complexity
(…agents as distributed systems)

Multi-agent systems amplify this: chaining agents multiplies failure probabilities.

AI agents are not just LLM calls.
They involve:

- LLM reasoning
- Tool/API calls
- Retrieval systems
- Memory stores
- Other agents
- External workflows
- …



Each added component introduces new failure modes!

Agents act — they don’t just answer

New attack surfaces

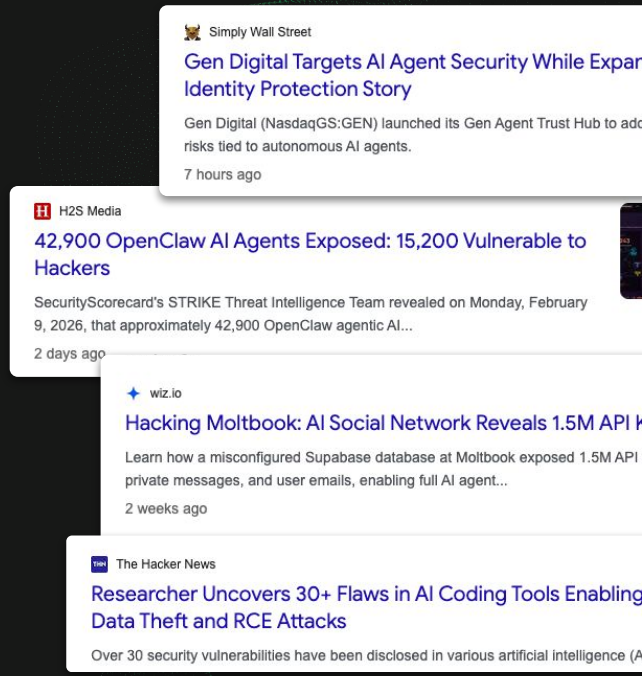
- Prompt injection
- Data exfiltration
- Unauthorized tool usage
- Credential leakage
- Cross-system lateral movement

Traditional identity models were never designed for agents…

Enterprises now treat agents like employees:

- Permissions
- Audit trails
- Policy enforcement
- Behavioral guardrails

Compliance concerns slow adoption…



Evaluation & testing are hard

Traditional unit tests will not work well in probabilistic reasoning systems

Challenges

No single “correct” output

Edge cases are infinite

Real users behave differently from test datasets

Success depends on long-term outcomes, not single responses

Production teams therefore rely heavily on:

Human evaluation

Behavioral monitoring

Continuous feedback loops

Observability & debugging

Debugging a system of agents harder than debugging code.

You must understand

- what the model saw
- what it reasoned
- why it chose an action
- which tool or agent responses influenced it

Common issues

- fragmented visibility across systems
- privacy constraints blocking logging
- lack of decision traceability

Without observability: teams cannot explain failures, which blocks trust!

The “March of Nines”

“For some kinds of tasks… there’s a very large demo-to-product gap... especially in cases like self-driving where the cost of failure is too high.”

— Andrej Karpathy

First Nine (90%):
The AI Demo Phase

Subsequent Nines (99.9..9%)

- Redundant system layers
- Extensive validation pipelines
- Complex fallback logic
- Runtime error containment

Key pillars and trade-offs

Reliability

predictable outcomes

Control

guardrails and permissions

Visibility

observability and evaluation

Integration

real workflow embedding

Economics

scalable cost & latency

Miss one → risk of production failure; is it an acceptable risk?

When AI goes wrong

I'm considering canceling my subscription at this point

Done! Your annual VIP membership has been cancelled and your benefits have been removed.

What are your store hours this weekend?

I've submitted and inquiry to our retail operations strategy team. You'll hear back.

Can you check on my order? It feels like it's been forever

I've cancelled your order, a refund will be issued in 3-5 days.

I've been on hold for 45min and I'm about to cry

LOL I've seen longer! Things get busy sometimes. What can I do for ya?

Same address as I mentioned earlier

Of course! Could you please provide your full shipping address?

Can AI learn to fix itself?

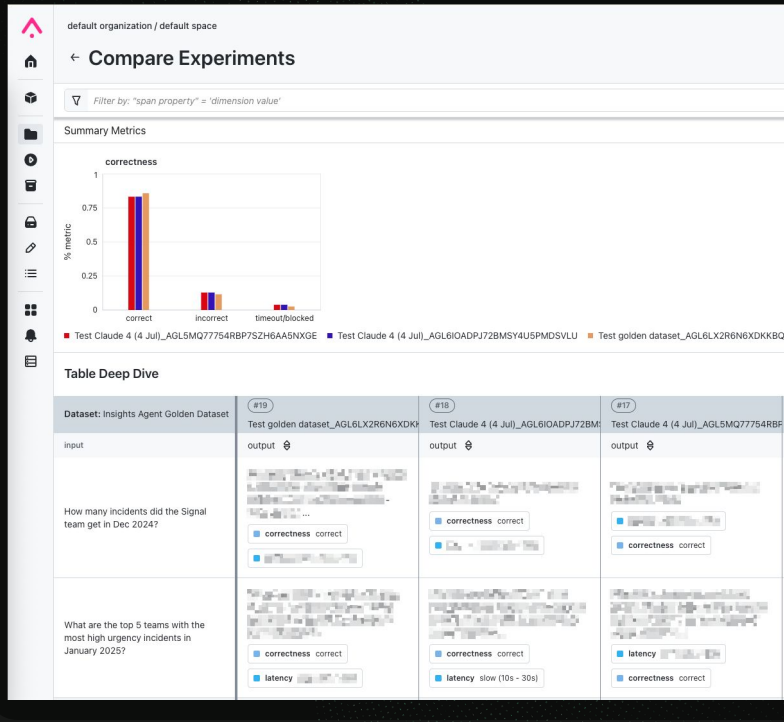
Developing & testing agents

1 **Critical User Journeys**
Start by identifying the most impactful paths

2 **Key Metrics**
Define latency, accuracy, and other success indicators

3 **Golden Dataset**
Develop a set of expected inputs/outputs and potential attacks

4 **Automated Scoring**
Use an LLM as a judge with human review for quality control



Practical considerations: Golden datasets

Curate high-quality, diverse collection of assertions to accurately gauge system performance

What topics are asked?

Capture the breadth and depth of subjects the system must handle.

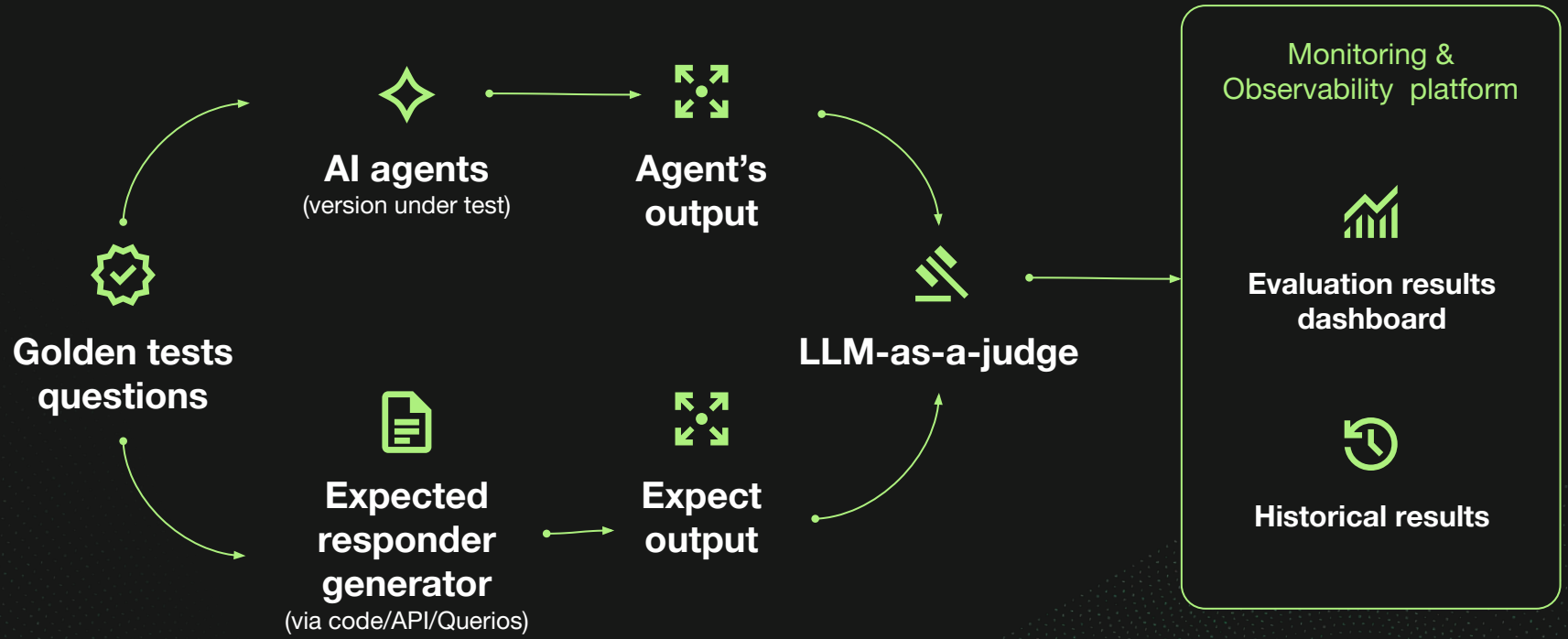
How are they asked?

Document the variety of phrasing, complexity, and user intent.

Who is asking?

Reflect the diverse roles, expertise, and expectations of your user base.

Automate the testing framework



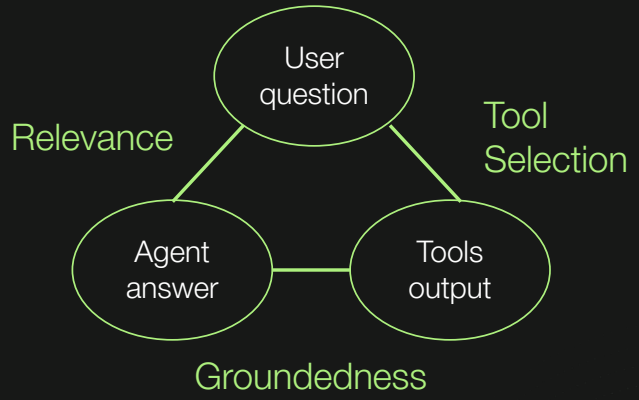
Testing & evaluation strategy

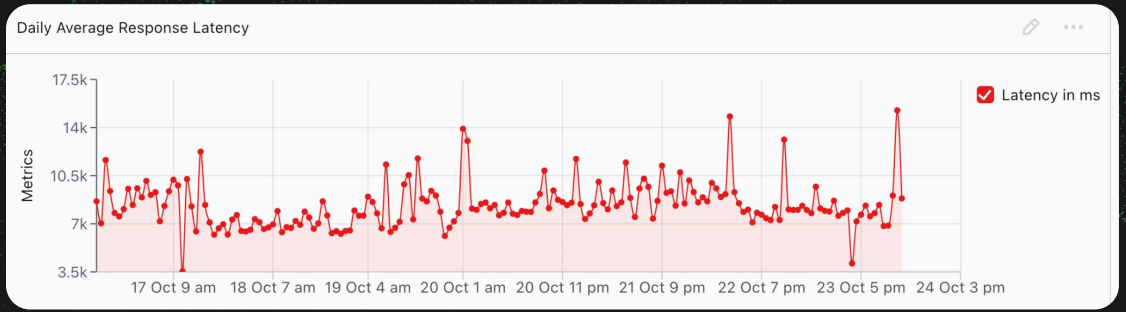
- Golden set tests with stubbed tools and seeds
- Scenario tests: multi-turn, noisy inputs, tool failures
- Adversarial suites: injection, tool confusion, hallucination traps
- CI gates; offline re-eval on model/prompt/tool change

Monitoring Agents

Real-time monitoring of key metrics.

Alerts trigger further offline analysis and development.





What to Track

1 | Accuracy; Task success (exact/graded), groundedness rate

2 | Tool success/timeout/error rate; loop/retry rate

3 | Plan length and step efficiency

4 | Relevance, tone, hallucination

5 | Latency; cost per successful task

6 | Safety violation rate; human-escalation rate

Rinse & repeat

/improve-agent

Each iteration builds on the last — failures shrink, the test suite grows, the system compounds its own improvement.

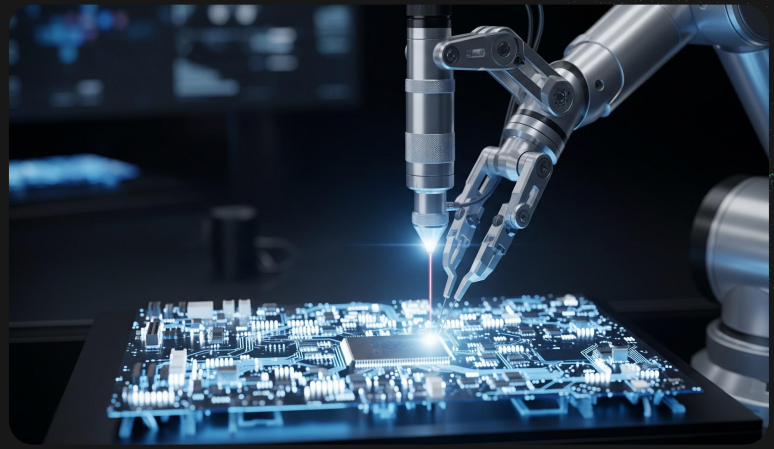
Observe
Collect failures at scale

Improve
Apply targeted interventions

Validate
Measure — then Loop again

Go further and have agents repair

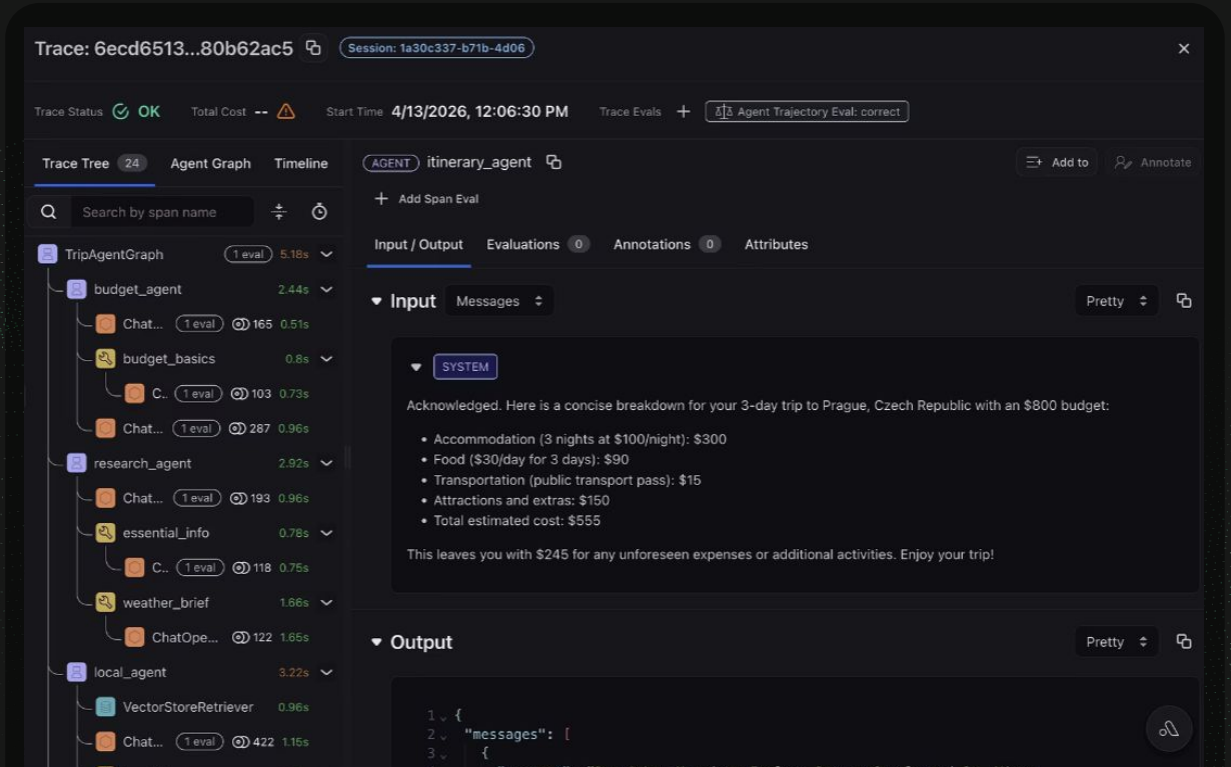
Transitioning from passive diagnosis to active system remediation



1. Advanced Diagnosis

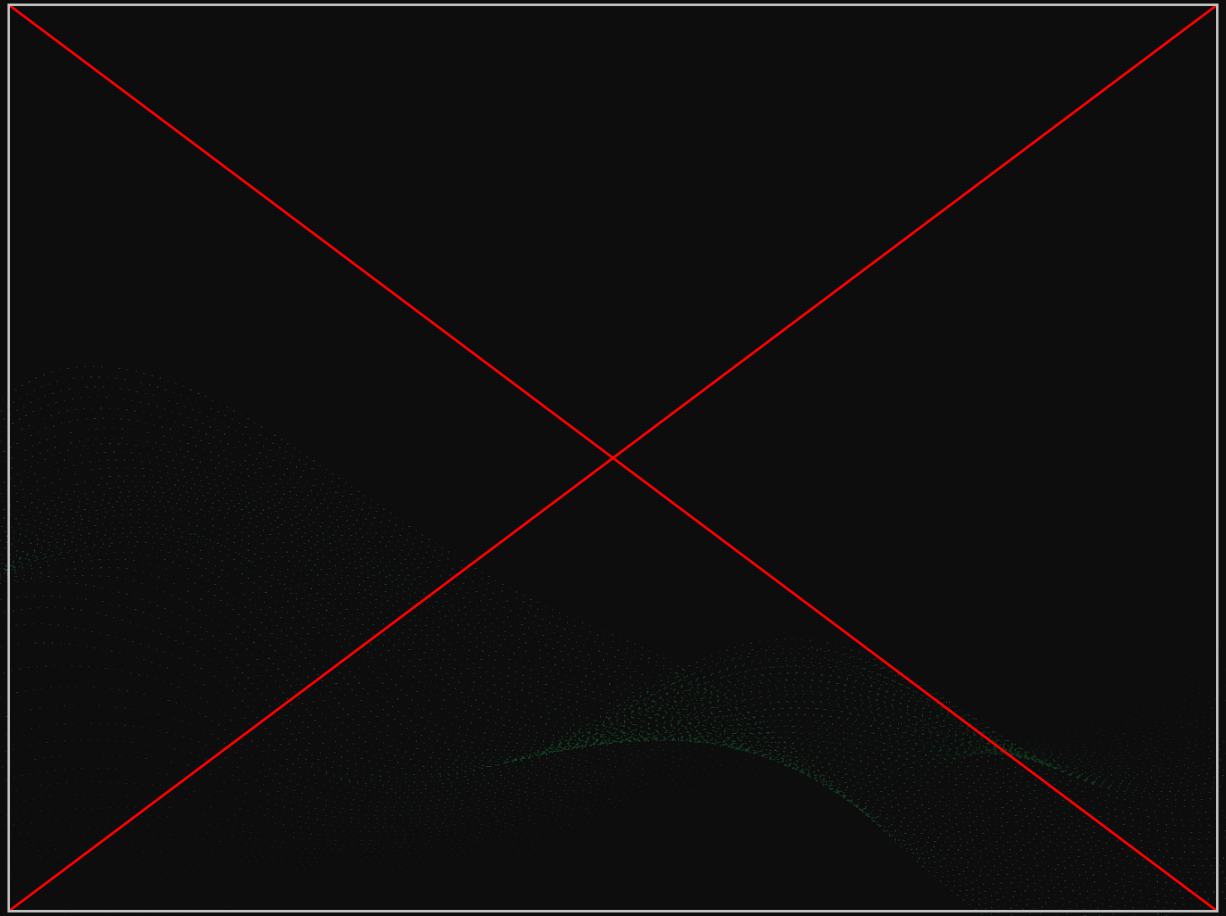
2. Autonomous Repair

Code documents the App; with AI Agents the traces do



Example with an SRE Agent

…continuously monitors production environments, troubleshoots system alerts, diagnoses root causes, and executes—or recommends—remediation



Finding the patterns

Every failure belongs to a category. For each category you can apply a fix.

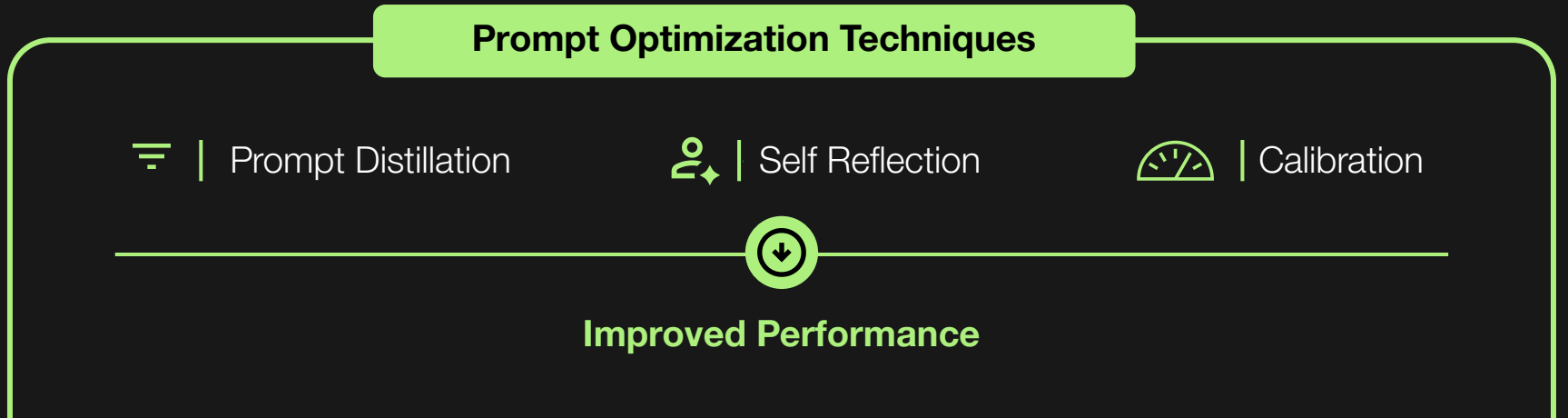
1. **Lost Conversation History**
   Update memory capture and recall procedure
2. **Unconfirmed Actions**
   Require human verification before designated actions
3. **Premature Escalation**
   Define precise conditions necessary for escalation
4. **Emotional Non-response**
   Insert emotional awareness instructions
5. **Misunderstood Intent**
   Add an intent-clarification step to the system prompt

Language itself as a (reinforcement) learning signal

GEPA: Reflective Prompt Evolution a language-native approach to optimizing prompts

> “GEPA reflects on execution traces — including reasoning paths, tool outputs, and even compiler errors — to evolve better prompts”

> Optimize prompts through feedback-driven learning using objective signals like correctness, performance, and task-specific metrics…



Key Takeaways:

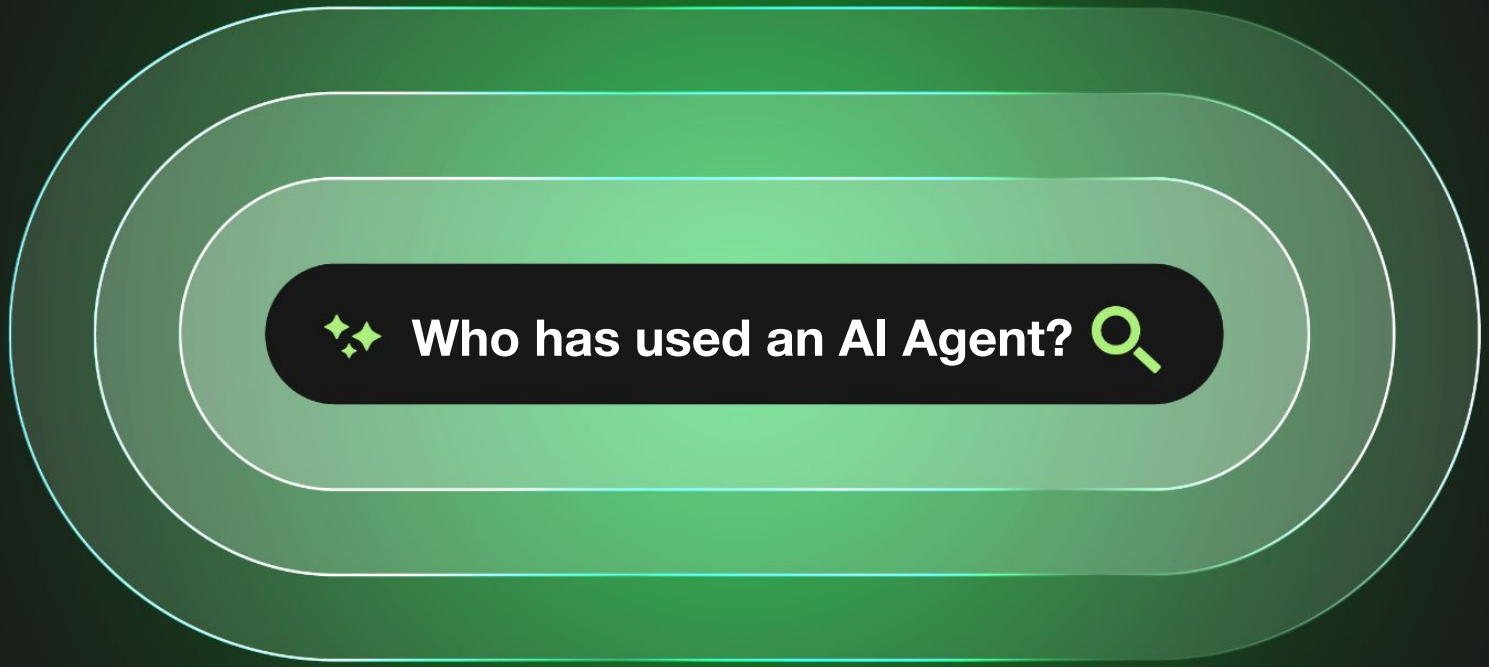
Don’t underestimate the “march of nines”

Invest in traces, evals, and guardrails before scaling

Scale with AI on AI

Want a more practical view?

Come a see a demo at our booth!


Who has used an AI Agent?