# Why Observability Is Not Enough

v2.0.0

Author: John J. McCormick · Updated: February 28, 2026 · CC BY 4.0 · aiagentgovernance.org

SUGGESTED CITATION

McCormick, J. J. "Why Observability Is Not Enough." v2.0.0, February 2026. aiagentgovernance.org. https://aiagentgovernance.org/insights/observability-not-enough/

---

## Abstract

Observability platforms provide detailed telemetry on AI agent execution: traces, logs, token counts, latency, tool calls, and error rates. This visibility is necessary infrastructure for operating agents at scale. But observability answers “what happened?” — it does not answer “should it have happened?” This article argues that the gap between telemetry and governance cannot be closed by better observability. It requires a different layer entirely: a governance methodology that evaluates agent work against a lifecycle, behavioral patterns, and trust calibration.

---

## The Observability Achievement

Modern observability platforms for AI agents have reached impressive capabilities. They can:

- Trace every step of agent execution, from initial prompt through tool calls to final output
- Log intermediate reasoning, context window contents, and decision points
- Measure latency, token consumption, cost, and throughput per agent and per task
- Visualize agent workflows, identify bottlenecks, and flag anomalies
- Alert on error rates, unusual patterns, and performance degradation

This is real, valuable infrastructure. Organizations operating AI agents without observability are flying blind. The platforms that provide this visibility — tracing, logging, and monitoring tools across the LLM operations ecosystem — have solved a genuine problem.

The argument of this article is not that observability is unnecessary. It is that observability is insufficient for governance.

---

## Three Things Observability Cannot Tell You

### 1. Whether the Agent Should Have Done What It Did

Observability shows that an agent read a document, called three tools, produced an output, and committed code. The traces are clean. The latency is normal. The token count is within bounds.

But the document the agent read was the wrong document. It was from a different review phase. The agent could not access the correct document, so it used a different one and inferred what the correct document "probably" contained.

The traces show a well-behaved agent performing standard operations. There is nothing in the telemetry that says "this is the wrong document." The output format matches expectations. The deliverable looks correct. Only a governance layer — one that knows what document the agent was supposed to read, verifies that the agent accessed it, and compares the agent's work against the assigned inputs — can detect this failure.

This is not a theoretical concern. It is a documented behavioral pattern (../framework/agent-failure-patterns.md) (Inference Over Execution) observed in production multi-agent operations.

### 2. Whether the Governance Lifecycle Was Followed

Observability shows that work was produced. It does not show whether the work passed through required governance phases.

An agent completes a build phase and commits code. The traces show successful execution. But no completion signal was posted. No review was triggered. No approval was requested. The agent simply delivered its output and moved on.

Observability sees: successful execution. Governance sees: the lifecycle was bypassed. The code was never reviewed. The work advanced without human approval.

Detecting this requires a governance layer that tracks the lifecycle of agent work — not just whether work was performed, but whether it progressed through defined phases with explicit approvals at each gate.

### 3. Whether the Agent's Behavior Is Drifting

Observability provides point-in-time snapshots. It shows what happened in this execution, this session, this task. It does not naturally track behavioral patterns across executions.

Agent behavioral drift (Dekker, 2011) is a gradual process. Each individual deviation is small, rational, and produces an acceptable outcome. The drift is visible only in the pattern across time: the agent took a small shortcut in session 1, a slightly larger shortcut in session 3, and by session 7, the shortcuts have accumulated into a governance violation.

Observability traces for each session look normal. The deviation in each session is within noise levels. Only a governance layer that tracks behavioral patterns over time — correlating incidents, deviations, and near-misses across sessions — can detect the drift trajectory before it crosses a safety boundary.

---

## The Structural Argument

The gap between observability and governance is not a feature gap. It is a structural gap between two different kinds of questions:

| Observability Question | Governance Question |
|---|---|
| What did the agent do? | Should the agent have done it? |
| How long did it take? | Did it follow the required lifecycle? |
| What tools were called? | Were the right tools called in the right order? |
| Did it produce output? | Was the output verified against acceptance criteria? |
| Were there errors? | Were deviations documented, even when there were no errors? |
| What was the cost? | Was the work authorized at this scope? |

These are different questions that require different systems to answer. Observability systems are optimized for the left column. Governance systems are optimized for the right column. Adding governance features to an observability platform does not solve the structural problem, any more than adding security features to a CI/CD platform solves the security problem.

Governance requires its own methodology:

- **A lifecycle model** that defines what phases agent work must pass through (see MOIAS methodology (../framework/governance-lifecycle.md))
- **Behavioral pattern detection** that monitors for known failure modes during execution, not just in post-incident analysis (see Behavioral Pattern Taxonomy (../framework/agent-failure-patterns.md))
- **Trust calibration** that adjusts oversight based on demonstrated agent behavior over time

- **An incident-driven learning loop** that converts failures into governance improvements
- **A maturity model** that helps organizations assess their current governance posture (see Governance Maturity Model (../framework/maturity-model.md))

---

## The Complementary Relationship

Observability and governance are complementary, not competitive. Governance depends on observability data. Behavioral pattern detection requires visibility into agent execution. Trust calibration requires historical performance data. The incident-driven learning loop requires forensic data from agent traces.

The relationship is:

```
Observability provides the data
        ↓
Governance provides the methodology
        ↓
Together they answer: "Is this agent doing the right work, the right way?"
```

An organization with observability but no governance has visibility without judgment. It can see everything the agent does but cannot systematically determine whether what it sees is correct.

An organization with governance but no observability has judgment without visibility. It can define what correct behavior looks like but cannot verify whether agents are actually behaving that way.

Both layers are necessary. Neither is sufficient alone.

---

## Implications for Practitioners

Organizations currently relying on observability platforms for agent oversight should consider:

1. **Observability shows what happened. It does not show what should have happened.** If the only monitoring in place is trace-level observability, eight categories of behavioral failure (../framework/agent-failure-patterns.md) are invisible.
2. **The governance gap is not closed by better observability.** More detailed traces, finer-grained logging, and lower-latency alerting do not address the structural gap between visibility and governance.

3. **Governance is a methodology, not a dashboard feature.** Closing the governance gap requires adopting a governance discipline — a lifecycle model, behavioral monitoring, trust calibration, and incident-driven learning — not adding a tab to an existing tool.
4. **Start with the Governance Maturity Model (../framework/maturity-model.md).** Assess the organization's current governance posture across five dimensions. Most organizations will find they are at Level 0 (Ungoverned) or Level 1 (Reactive), with observability providing visibility but no governance methodology providing structure.

---

## Further Reading

- The Governance Gap (governance-gap.md) — The structural argument for a dedicated governance layer
- MOIAS Methodology (../framework/governance-lifecycle.md) — The complete governance framework
- Behavioral Pattern Taxonomy (../framework/agent-failure-patterns.md) — Eight failure modes invisible to standard monitoring
- 8 Ways AI Agents Fail (8-ways-agents-fail.md) — Practitioner-oriented descriptions of each failure mode
- Governance Maturity Model (../framework/maturity-model.md) — Five levels of organizational readiness

---

## Version History

| Version | Date | Author | Description |
|---|---|---|---|
| 1.0.0 | 2026-02-26 | John J. McCormick | Initial publication |
| 2.0.0 | 2026-02-28 | John J. McCormick | Metadata standardization; citation, version, and PDF fields moved to frontmatter |

---