Nick Clark · Published April 25, 2026 · PDF

How to Detect When an AI Agent Contradicts Its Own Prior Decisions

If you run an autonomous agent long enough, it will eventually contradict a decision it made earlier: approve what it once refused, relax a standard it once enforced, or quietly reverse a commitment nobody re-approved. This guide describes an architectural approach for catching those self-contradictions as they happen, by scoring each new decision against the pattern the agent has already established. The approach is disclosed in United States Patent Application 19/647,395 and belongs to its Integrity and Coherence inventive step. It is an architecture you build yourself, not a shipping library.

What You Are Building

You are building a mechanism that answers one question at decision time: *does this action contradict the pattern this agent has already established?*

This is the problem behind the search query. A long-running agent accumulates a history of choices, and that history implies a stance: quality standards it holds itself to, commitments it has honored, refusals it has issued. Nothing in a typical LLM loop remembers that stance. Each turn is evaluated more or less fresh, so the agent can silently reverse itself, and the reversal only surfaces when a human notices the two decisions side by side.

The people who have this problem are the ones running agents that act over time and across sessions: coding agents that approve a change they rejected an hour ago, procurement or approval agents that green-light what they previously blocked, support agents that promise what policy forbids. What you want is not a post-hoc audit but an in-line check that flags the contradiction *before* the second decision commits, and leaves an auditable record of why it was flagged.

The architecture below, disclosed in United States Patent Application 19/647,395, treats consistency as an accumulated, continuously updated property of the agent rather than a per-request test.

Why the Obvious Approaches Fall Short

The intuitive fixes each address part of the problem and miss the structural piece.

Prompting the model to "be consistent." You can paste prior decisions into context and ask the model to reconcile them. This works until the relevant prior decision falls out of the window, or until there are hundreds of them and no principled way to select which matter. Consistency becomes a function of what happened to be in context, not of the agent's actual history.

Logging decisions and auditing later. Structured logs are necessary, but a log read after the fact catches contradictions only once a human or batch job goes looking. The contradictory action has already committed. Logging tells you what happened; it does not gate what is about to happen.

Rule engines and policy checks. A static rule ("never approve X") catches violations of *fixed* policy. Self-contradiction is different: the agent may be violating a standard *it* established through its own past behavior, one no author wrote down. There is no rule to match against because the norm is emergent, encoded in the trajectory of prior choices rather than in a policy file.

The common gap is that none of these carry a running, comparable representation of the agent's own normative pattern that a new decision can be measured against. That representation, and the comparison, is what the disclosed architecture supplies.

The Architecture

The disclosed approach rests on four elements: a persistent integrity field, an integrity engine that evaluates actions against declared values, a semantic-dissonance metric that measures divergence, and a lineage record that makes the whole thing reconstructible. All four are described in the filing.

An integrity field carried by the agent. The agent carries an integrity field as a first-class cognitive domain, tracked with a current value and a trajectory over time. Per the filing the field is structured across three domains: personal (alignment with the agent's own declared standards), interpersonal (honoring commitments to others), and global (consistency with broader system norms). A weighting module combines these into a composite integrity score using domain weights specified by policy, deterministically, not negotiated by the agent. Crucially, the integrity computation is performed by the agent's own integrity engine as a first-class operation; external systems may audit the field, but the score is computed from the agent's history, not imposed per request.

The integrity trajectory as the baseline. The lineage field records the complete history of the agent's state evolution: every mutation, delegation, and governance decision. The integrity engine reads that lineage as its evidentiary basis, evaluates the recorded actions against the agent's declared values, and writes the result back into the lineage. The filing calls the accumulated pattern of those evaluations the agent's **integrity trajectory**. This is the "established pattern" a new decision gets checked against. It is not a snapshot; it is a direction and rate of change over recent evaluation windows.

Semantic dissonance as the contradiction signal. This is the mechanism most directly aimed at the search query. The filing describes *semantic dissonance logging*: the recording of conditions in which the agent's actions produce inconsistency with its own declared operational narrative. Semantic dissonance is computed as a distance metric between the agent's *actual behavioral vector* (derived from the lineage) and its *declared behavioral vector* (derived from the intent field and declared value set). When that distance exceeds a policy-defined threshold, the integrity engine records a dissonance event: a lineage entry identifying the specific dimensions of inconsistency, the magnitude of the divergence, and whether the dissonance is increasing, stable, or decreasing. That is a concrete, machine-detectable definition of "the agent is contradicting its prior decisions."

Prospective gating, not just accounting. The filing is explicit that every mutation to the agent's state, whether proposed by an external inference engine, generated by the agent's own forecasting, or inherited through delegation, is evaluated against the integrity model *before commitment*. The integrity engine computes the projected impact of a proposed mutation on each integrity domain. If the mutation would push the composite integrity score below a policy-defined threshold, it is flagged for enhanced scrutiny, and a governance gate receives the integrity impact assessment as an additional input to its admissibility decision. So the contradiction check runs as a prospective filter on the pending decision, not only as a retrospective audit.

Lineage makes it reconstructible. Because deviation and dissonance events are themselves recorded as lineage entries, the filing describes the deviation log as an indexed, queryable view over the lineage optimized for audit and trajectory analysis. Each entry carries enough detail (identifiers, timestamps, affected domains, severity classification, the divergence dimensions) to reconstruct why a contradiction was flagged and to replay it later. Detection and explainability come from the same record.

How to Approach the Build

You are implementing this yourself. The steps below follow the architecture; the interface sketch is illustrative and faithful to the filing, not a package you can install.

1. **Give the agent a persistent state object with a lineage.** Every decision the agent commits must append an immutable entry describing the action, the context, and the declared intent under which it acted. Without a durable lineage there is no trajectory to compare against. Persist it across sessions, not just within one.
2. **Define the declared behavioral vector.** Decide, per your domain, what "the agent's declared values" concretely means: quality standards, commitments, refusal conditions, policy commitments. This is the yardstick. The dissonance metric is only as meaningful as this definition, so make it explicit and versioned.
3. **Derive the actual behavioral vector from lineage.** Reduce the recorded history into a comparable representation of how the agent has actually been behaving. This is where most engineering effort goes: choosing an embedding or feature encoding for past decisions such that "approve this class of change" and "reject this class of change" land at a measurable distance.
4. **Implement the integrity engine as an evaluation over the two vectors.** For a pending decision, compute the distance between the actual and declared behavioral vectors, and separately the projected impact of committing this decision on each integrity domain. Illustrative interface only:

```
// illustrative, spec-faithful sketch, not a library
dissonance = distance(actual_vector(lineage), declared_vector(intent, valu
impact     = project_integrity_impact(pending_decision)  // per domain
if dissonance > policy.dissonance_threshold
     or composite(impact) < policy.integrity_floor:
    record_dissonance_event(lineage, dimensions, magnitude, trajectory)
    gate.flag_for_scrutiny(pending_decision, impact)
```

5. **Wire it into a governance gate before commit.** The check must run on the *proposed* mutation, before it takes effect, and hand its assessment to whatever admissibility decision you already have. A flag can mean escalate to a human, block, or require re-justification, per your policy.
6. **Record the outcome back into lineage.** Whether flagged or cleared, write the evaluation result as a lineage entry. That is what makes the next comparison richer and what makes any flag replayable and explainable after the fact.
7. **Track the trajectory, not just point events.** Compute the integrity score over recent windows so you can see whether dissonance is increasing, stable, or decreasing. A single reversal may be legitimate; a rising trend of them is the signal the architecture is built to surface.

What This Does Not Give You

This is an architecture, not a drop-in library, and not a benchmarked or productized system. There is nothing to `npm install` here; you build every element above yourself, and its behavior depends entirely on choices the filing leaves to you.

The hardest of those choices is the behavioral-vector encoding and the distance metric. The filing establishes *that* dissonance is a distance between actual and declared vectors and *that* it is compared to a policy-defined threshold; it does not hand you the embedding, the threshold values, or accuracy numbers, and this guide invents none. Get the encoding wrong and you get false contradictions on legitimate context-dependent decisions, or you miss real ones. Expect to tune the threshold against your own traffic.

It also will not adjudicate whether a reversal is *correct*. Detecting that today's decision contradicts the established pattern is not the same as knowing which of the two is right; a changed policy or new evidence can make a contradiction the desirable outcome. The

architecture surfaces and records the divergence and routes it to a governance decision. It does not replace that decision.

Finally, it applies where the agent has a durable, honest history to reason over. An agent with no persisted lineage, or one whose declared values are never defined, has no trajectory to be measured against, and the mechanism has nothing to compare.

Disclosure Scope

The architecture described in this guide, including the integrity field, the integrity engine, the accumulated integrity trajectory, semantic-dissonance detection, and pre-commit integrity gating recorded in lineage, is disclosed in United States Patent Application 19/647,395. This guide is educational: it explains an approach a developer can implement independently. It is not a warranty, a specification of a shipping product, or an offer of software, and it does not guarantee any particular result. Every design parameter not stated in the filing is left to the implementer.

Integrity & Coherence (/integrity-coherence)

All 40 steps → (/inventive-steps)

Track normative consistency. Detect deviation. Self-correct.

Chapter 3 (/patents/19-647395/chapters/integrity)

PRIMARY TECHNICAL DISCLOSURE

- The Coherence Trifecta: Empathy, Integrity, and Self-Esteem as a Unified Control Loop (/articles/the-coherence-trifecta-empathy-self-esteem-and-integrity-as-a-unified-control-loop)

SECONDARY TECHNICAL

- Three-Domain Integrity Model (/articles/integrity-coherence/three-domain-model)
- Deviation Function D=(N-T)/(ExS) (/articles/integrity-coherence/deviation-function)
- Self-Esteem as Internal Validator (/articles/integrity-coherence/self-esteem-validator)

- Deviation as Deterministic Semantic Mutation (/articles/integrity-coherence/deviation-mutation)
- Integrity Structural Placement (/articles/integrity-coherence/structural-placement)
- Empathy as Distributed Moral Load (/articles/integrity-coherence/empathy-mechanism)
- Coherence Trifecta Control Loop (/articles/integrity-coherence/coherence-trifecta)
- Coping Intercept Patterns (/articles/integrity-coherence/coping-intercepts)
- Integrity Deviation Logging (/articles/integrity-coherence/deviation-logging)
- Integrity Collapse Detection (/articles/integrity-coherence/collapse-detection)
- Redemption Engine (/articles/integrity-coherence/redemption-engine)
- Moral Trajectory Forecasting (/articles/integrity-coherence/moral-trajectory)
- Integrity-Aware Trust Slope Validation (/articles/integrity-coherence/trust-slope-integrity)
- Integrity-Confidence Cross-Primitive Coupling (/articles/integrity-coherence/confidence-coupling)
- Integrity-Modulated Discovery Traversal (/articles/integrity-coherence/discovery-integrity)
- Integrity-Aware Multi-Agent Negotiation (/articles/integrity-coherence/multi-agent-negotiation)
- Biological Signal Coupling for Integrity (/articles/integrity-coherence/biological-integrity)
- Policy-Based Integrity Constraints (/articles/integrity-coherence/policy-constraints)
- Integrity Field Portability (/articles/integrity-coherence/field-portability)
- Predictive Deviation Alerting (/articles/integrity-coherence/predictive-alerting)
- Governed Forgetting (/articles/integrity-coherence/governed-forgetting)
- Predictive Social Modeling (/articles/integrity-coherence/predictive-social-modeling)
- Refusal as First-Class Observation (/articles/integrity-coherence/refusal-as-observation)
- Historical Policy-Version Reconstruction (/articles/integrity-coherence/historical-policy-version-reconstruction)

APPLICATIONS · GENERAL

- Autonomous Vehicle Ethical Decision-Making Through Computable Integrity (/articles/integrity-coherence/autonomous-vehicle-ethics)
- Detecting Strategy Drift in Algorithmic Trading Agents With Computable Integrity (/articles/integrity-coherence/trading-normative-consistency)
- How to Keep a Legal AI Agent's Advice Consistent With Precedent and Its Own Prior Positions (/articles/integrity-coherence/legal-advisory-agents)
- Government AI Policy Agents: Consistency, Equity, and Statutory Alignment by Design (/articles/integrity-coherence/government-policy-agents)

- AI Editorial Agents for Newsrooms: Consistent Standards and Bias-Drift Detection by Design (/articles/integrity-coherence/journalism-editorial-agents)
- Integrity and Coherence for AI Environmental Compliance Agents: Consistent Regulatory Interpretation Across Facilities and Jurisdictions (/articles/integrity-coherence/environmental-compliance)
- Integrity and Coherence for Insurance Underwriting Agents (/articles/integrity-coherence/insurance-underwriting)
- Integrity and Coherence for Social Media Moderation Agents (/articles/integrity-coherence/social-media-moderation)
- Legal-Evidence Reconstruction for Autonomous-Incident Litigation: Court-Admissible Policy-Version Lineage (/articles/integrity-coherence/legal-evidence-reconstruction)
- Regulatory Audit Replay With Historical Policy Versions (/articles/integrity-coherence/regulatory-audit-replay)

APPLICATIONS · SPECIFIC

- Waymo vs a Governed Integrity Layer for Autonomous Behavior (/articles/integrity-coherence/waymo)
- Cruise vs Governed Autonomy: Why AV Safety Needs a Computable Integrity Field (/articles/integrity-coherence/cruise)
- JPMorgan Trading Compliance vs Governed Agents: The Integrity Field Gap (/articles/integrity-coherence/jpmorgan)
- Palantir Governance Alternative: Adding a Computable Integrity Field to Government Analytics (/articles/integrity-coherence/palantir)
- Does the Aurora Driver maintain normative memory across decisions? (/articles/integrity-coherence/aurora-innovation)
- Nuro Alternative: Governed Autonomous Delivery Beyond Per-Trip Safety Records (/articles/integrity-coherence/nuro)
- Zoox vs Governed Autonomy: Tracking Normative Drift the Planner Cannot See (/articles/integrity-coherence/zoox)
- Motional Alternative for Governed Normative Trajectory in Autonomous Driving (/articles/integrity-coherence/motional)
- Argo AI Legacy vs Governed Autonomy: The Missing Deviation Function (/articles/integrity-coherence/argo-ai-legacy)
- comma.ai openpilot and the Governed-Behavior Layer: Integrity Coherence for Learning-Based Driving (/articles/integrity-coherence/comma-ai)
- Apache Iceberg Time Travel vs Governed Replay: Data Without Policy (/articles/integrity-coherence/apache-iceberg-time-travel)

Integrity & Coherence overview → (/integrity-coherence)