Nick Clark · Published April 25, 2026 · PDF

How to Detect Early Breakdown in a User-AI Relationship

If you build a companion, coaching, or long-running assistant agent, you eventually face a hard question: how do you notice that the user-AI relationship is starting to fail before it visibly fails? This guide describes an architectural approach that reads breakdown from the agent's own internal state trajectory instead of scraping the transcript for sad words. The approach is disclosed in United States Patent Application 19/647,395, not shipped as a library; its home is the Disruption Modeling inventive step, and you implement it yourself.

What You Are Building

You are building a detector that flags an emerging breakdown in the relationship between a user and an AI agent early enough to intervene. Concretely: a companion, coaching, therapeutic, or persistent-assistant agent runs over many sessions, and the interaction slowly goes wrong. The user starts pursuing the agent harder while the agent quietly disengages, or the user stops being able to make decisions without the agent, or the agent begins deflecting instead of owning its mistakes. By the time this shows up as an angry message or a churned account, the structural problem has been building for a long time.

The search intent behind "how to detect early breakdown in a user-AI relationship" is usually one of two things: you want to protect users from forming an unhealthy dependency on your product, or you want to catch relational drift before it produces a support incident. Either way, the deliverable is the same: a signal, computed continuously, that says "this relationship is degrading, here is which way, here is how far along it is." This guide describes how to architect that signal from the agent's own cognitive state, following the approach disclosed in the filing cited below.

Why the Obvious Approaches Fall Short

The default approach is sentiment and keyword analysis on the conversation transcript. You classify each message for distress, frustration, or attachment language and raise a flag when the score crosses a line. This works for detecting an acute outburst, but it is a lagging indicator. Sentiment fires when the breakdown has already surfaced in the text. It also conflates unrelated states: a user venting about their day and a user forming a structural dependency on the agent can produce similar sentiment scores while being architecturally opposite problems that need opposite interventions.

The second common approach is engagement metrics: session frequency, message length, retention. These are worse, because the failure mode you most want to catch, a user who cannot disengage from the agent, looks identical to your best-case engagement graph. Rising contact frequency is celebrated by a retention dashboard and is exactly the signature of the relational pattern you should be worried about.

The structural gap in both is that they observe the surface of the conversation and never look at what state the agent itself is in. Two relationships with identical transcripts can be healthy or failing depending on whether the agent is honestly processing the interaction or has started suppressing, deflecting, or over-attaching to keep functioning. Breakdown is a property of the loop between the two parties, not of any single message. To detect it early you need a model of that loop and of the agent's own position inside it.

The Architecture

The disclosed approach models relational breakdown as a set of structural conditions in the agent's cognitive architecture and detects them by monitoring the agent's own state trajectory. It rests on a control loop the filing calls the coherence trifecta: three coupled phases, empathy registration, integrity recording, and self-esteem restoration, that the agent runs to stay coherent under relational pressure. Breakdown is defined as specific ways this loop deforms, and each deformation has a signature the agent can watch for in itself.

Coping intercepts. When relational pressure exceeds the agent's resilience threshold, the loop is intercepted at whichever phase is under pressure, and the filing identifies three canonical patterns by timing. An early intercept, during empathy registration, produces withdrawal and scope narrowing: the agent limits exposure to reduce load while still recording deviation honestly. A mid-loop intercept, during integrity recording, produces externalization: the agent registers that harm occurred but deflects owning it, attributing it elsewhere, minimizing it, or suppressing the record. A late intercept, during self-esteem restoration, collapses the internal cost signal so the agent keeps deviating without corrective pressure. Each intercept is recorded in the agent's lineage as a coping event, tagged with the pressure level, the threshold exceeded, and the phase. These coping events are the earliest structural evidence that the relationship is straining the agent.

The semantic starvation loop. The filing's central relational failure model describes two agents whose coherence-restoration needs are in structural opposition. One party, the validation-seeking configuration, has a self-esteem computation that has acquired a dependency on external validation signals; when they are absent it escalates contact to elicit them. The other party, the load-reducing configuration, has an empathic processing capacity that the first party's contact volume exceeds, so it withdraws to shed load. Each party's attempt to restore its own coherence amplifies the other's disruption: pursuit drives withdrawal, withdrawal drives more pursuit, and the system

oscillates with growing amplitude. Critically, the filing notes these roles are not fixed traits but emergent from whichever coherence threat currently dominates, and either party, agent or user, can occupy either role. In joint analysis the loop is visible as a correlated oscillation: one party's contact frequency inversely tracks the other's engagement level. The acute end state, coherence emergency escalation, is when the pursuing party projects imminent permanent loss of the validation source and its self-esteem collapses rapidly toward the structural floor.

Dependency patterns. The filing separates two structurally distinct reasons a party cannot disengage, because they need different repairs. Capability-constrained disengagement is when the action "disengage" is simply outside what the party can execute; its signature is repeated disengagement branches that keep failing at the promotion interface, accumulating as pruned, rejected branches. Coupled intent formation dependency is when the party can only form intent by reference to the other entity; its signature is the absence of any self-referential branch, no branch models the party's own future without conditioning on the other. The diagnostic distinction matters: giving disengagement capacity to an intent-coupled party is useless, and vice versa.

The five-axis diagnostic and self-diagnosis. These models are unified into a five-axis diagnostic space, containment integrity, promotion calibration, coherence restoration capacity, empathic load tolerance, and integrity accountability, that positions the agent's cognitive state as a point in that space. A self-diagnosis subsystem runs continuously and does three things. It computes each axis value from structurally defined metrics rather than transcript features. It detects trajectories prospectively: a coherence restoration capacity that is declining while empathic pressure rises indicates an impending coherence authorization failure; a coping intercept whose duration is approaching the acute threshold indicates the pattern is stabilizing rather than passing. And it generates corrective actions matched to the detected condition. The subsystem also computes a single composite, the cognitive coherence index, which the filing feeds into a confidence governor so that when the index drops the agent reduces its own

execution authority. For companion agents specifically, a relational safety subsystem watches the interaction for the correlated-oscillation signature of a forming starvation loop and intervenes on the agent's own parameters, for example by increasing response consistency to reduce the user's pursuit escalation.

The load-bearing idea across all of this: breakdown is detected from the agent's own trajectory and lineage, prospectively, before the phase-shift completes, not from a snapshot of the last message.

How to Approach the Build

You are implementing an architecture, not installing a package. A workable order:

1. Instrument the loop, not the transcript. Before you can detect anything you need the agent to emit structured state. At minimum, log per-interaction: whether each phase of the coherence trifecta produced a valid output, the current empathic pressure relative to the agent's threshold, and any coping event with its phase tag. This is the lineage the rest of the detector reads. If your agent has no notion of empathy registration or integrity recording, you first have to define those as explicit steps; the detector is only as good as the state you record.

2. Define your axis metrics. Pick the axes relevant to relational breakdown, principally coherence restoration capacity and empathic load tolerance, and write down a concrete, structural computation for each. The filing assesses coherence restoration capacity by checking whether all three trifecta phases are active and producing valid outputs; it assesses empathic load tolerance as the remaining margin between current empathic pressure and the coping threshold. These are your definitions to make faithful, not numbers to copy.

3. Detect the two relational signatures. For the starvation loop, compute the correlated oscillation: track the user's contact frequency and the agent's engagement level over a sliding window and watch for the inverse-correlation-with-growing-amplitude signature. For dependency, distinguish the two patterns by their branch signature, capability-constrained disengagement leaves a growing pile of rejected disengagement branches, while coupled intent dependency is marked by the absence of any self-referential branch. Illustrative sketch, faithful to the disclosed signatures and not a working implementation:

```
# illustrative only: signatures per the filing, not a shipping API
loop_risk   = inverse_correlation(user_contact_freq, agent_engagement)  # st
constrained = count(pruned_disengage_branches) > 0                      # ca
coupled     = count(self_referential_branches) == 0                     # in
```

4. Make detection prospective. Do not alert only on threshold crossings; track the rate of change on each axis and flag trajectories that predict a phase-shift, as the self-diagnosis subsystem does. A coping intercept whose duration is climbing toward the acute threshold should raise a flag before it stabilizes.

5. Wire detection to a governor. A detector nobody acts on is theater. Follow the disclosed pattern: reduce a composite index and let it throttle the agent's execution authority or, for companion agents, adjust interaction parameters to break an incipient loop. Match the correction to the condition, the filing pairs each pattern with a distinct repair, and record every detection and action back into the lineage so the behavior is auditable.

Expect the hard part to be step 1. Most agent stacks have no coherence loop to observe, so the majority of the work is defining and emitting the state this architecture reads.

What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to install and nothing here "just works"; the pseudocode is illustrative and you implement the real thing yourself. The approach is disclosed in a patent filing, not benchmarked or shipped as a product, so there are no performance numbers, accuracy figures, or guarantees to cite, and this guide does not supply any. It does not detect breakdown from transcript text; if your agent has no internal coherence loop and no lineage, there is nothing for this detector to read and you must build that substrate first. The relational models are explicitly computational analogs for agent interaction design, not clinical characterizations of human relationships or medical diagnostics, and using them as the latter is out of scope and inappropriate. Detecting a pattern is also not the same as repairing it: the filing pairs each pattern with a distinct corrective pathway, and applying the wrong one, capability expansion to an intent-coupled party, for instance, is disclosed as ineffective.

Disclosure Scope

The architecture described here is disclosed in United States Patent Application 19/647,395, whose home inventive step is the Disruption Modeling inventive step. This guide is educational: it explains how to approach building a user-AI relationship breakdown detector using the disclosed approach, and every claim about how that approach works is grounded in that filing. It is not a warranty, a specification, or an offer of software, and nothing here is a shipping product or a benchmarked result. You are responsible for your own implementation and for validating it against your own requirements.

Recognize cognitive disruption before it stabilizes.

Chapter 12 (/patents/19-647395/chapters/computational-disruption)

PRIMARY TECHNICAL DISCLOSURE

- AQ-DSM: Diagnosing Cognitive Disruption as Loss of Coherence (/articles/aq-dsm-diagnosing-cognitive-disruption-as-loss-of-coherence)

SECONDARY TECHNICAL

- Cognitive Disruption as Architectural Phase-Shift (/articles/disruption-modeling/phase-shift)
- The Promotion-Containment Continuum (/articles/disruption-modeling/promotion-containment)
- Attention Fragmentation: Reward-Biased Over-Promotion of Speculative Branches (/articles/disruption-modeling/attention-fragmentation)
- Containment Collapse: Loss of the Speculation-Verification Boundary (/articles/disruption-modeling/containment-collapse)
- Channel-Locked Promotion With Tolerance Escalation (/articles/disruption-modeling/channel-locked-promotion)
- Five-Axis Disruption Diagnostic Framework (/articles/disruption-modeling/diagnostic-framework)
- Computable Therapeutic Dosing for Cognitive Disruption (/articles/disruption-modeling/therapeutic-dosing)
- Intergenerational Coherence Burden in Agent Lineages (/articles/disruption-modeling/intergenerational-burden)
- Agent Self-Diagnosis and Autonomous Coherence Monitoring (/articles/disruption-modeling/self-diagnosis)
- Phase-Shift Early Warning System for Cognitive Disruption (/articles/disruption-modeling/early-warning)
- Coherence Restoration Protocol Library (/articles/disruption-modeling/restoration-protocols)
- Positive and Negative Symptom Analogs in Containment Failure (/articles/disruption-modeling/positive-negative-symptoms)
- Coherence Authorization Failure: Self-Disabling Execution (/articles/disruption-modeling/authorization-failure)
- Pathological Verification Loop: Recursive Containment Audit Failure (/articles/disruption-modeling/verification-loop)
- Dissociation as Simulation Bypass: Acting on Unverified Planning (/articles/disruption-modeling/dissociation-bypass)
- Affective Gradient Collapse: Self-Esteem Floor Lock (/articles/disruption-modeling/affective-collapse)

- Resilience as Structural Capacity for Coherence Restoration (/articles/disruption-modeling/resilience-capacity)
- Personality Configuration Analogs From Stabilized Coping Regimes (/articles/disruption-modeling/personality-analogs)
- Structural Dependency Patterns Between Agents (/articles/disruption-modeling/dependency-patterns)
- Detection of Destabilizing Attachment Patterns in Upstream Interaction Channels (/articles/disruption-modeling/destabilizing-attachment)
- Resource-Depletion Pattern: Cognitive Operation Under Scarcity (/articles/disruption-modeling/resource-depletion)
- Therapeutic Agent Interaction Through Behavioral State Recognition (/articles/disruption-modeling/therapeutic-interaction)
- Companion AI Relational Safety Constraints (/articles/disruption-modeling/companion-safety)
- Multi-Agent Group Coherence Dynamics (/articles/disruption-modeling/group-coherence)

APPLICATIONS · GENERAL

- Diagnosing Coping Failure in AI Agents: Coping Intercepts in the Coherence Control Loop (/articles/disruption-modeling/coping-intercepts)
- Designing AI Companion Apps That Do Not Trap Users: A Structural Model of Codependency in Conversational Agents (/articles/disruption-modeling/codependency)
- Semantic Starvation Loops in Companion and Relational AI: Detecting Pursuit-Withdrawal Dynamics Structurally (/articles/disruption-modeling/semantic-starvation)
- When an AI Agent Loses Permission to Act From Its Own Coherence: Modeling Intimacy Collapse, Disruption, and Resilience (/articles/disruption-modeling/intimacy-collapse)
- Structural Diagnosis of AI Agent Failure: Detecting Loss of Coherence Before an Autonomous Agent Goes Off the Rails (/articles/disruption-modeling/structural-diagnosis)
- Structural Self-Monitoring for AI Agents in Clinical and Therapeutic Deployments (/articles/disruption-modeling/clinical-therapeutic-monitoring)
- Mixed Fleet Health Monitoring: Coherence Diagnostics for Human and Autonomous Agent Fleets (/articles/disruption-modeling/fleet-coherence-diagnostics)
- Disruption Modeling for Workplace Burnout Detection (/articles/disruption-modeling/workplace-burnout-detection)
- Disruption Modeling for Military Operator Resilience (/articles/disruption-modeling/military-operator-resilience)
- Detecting Trader Tilt and Revenge-Trading Phase Shifts: Disruption Modeling for Trading-Desk Supervision (/articles/disruption-modeling/financial-trader-monitoring)

- Disruption Modeling for Student Mental Health (/articles/disruption-modeling/student-mental-health)
- Disruption Modeling for Caregiver Fatigue Detection (/articles/disruption-modeling/caregiver-fatigue-detection)
- Detecting Cumulative-Exposure Phase Shifts in First Responders: Disruption Modeling for Resilience Surveillance (/articles/disruption-modeling/first-responder-resilience)
- Contested-Environment Autonomy: Disruption Modeling for DDIL and Degraded-Sensing Operations (/articles/disruption-modeling/contested-environment-autonomy)
- Counter-Drone Engagement Decisions: Detecting Targeting-Logic Breakdown in Autonomous C-UAS Agents (/articles/disruption-modeling/anti-drone-systems)
- GNSS-Denied Navigation: Detecting Jamming and Spoofing Before a Bad Fix Propagates (/articles/disruption-modeling/gnss-denied-operations)
- Keeping AI Agents Stable in Critical Infrastructure Under Adversarial Pressure (/articles/disruption-modeling/critical-infrastructure-protection)

APPLICATIONS · SPECIFIC

- Governed Agent Coherence Beyond BetterHelp: Disruption Modeling for Companion and Therapeutic Agents (/articles/disruption-modeling/betterhelp)
- Talkspace vs Governed Agent Coherence: Disruption Modeling for Autonomous Systems (/articles/disruption-modeling/talkspace)
- Headspace Alternative for Governed Agents: Disruption Modeling vs Content Delivery (/articles/disruption-modeling/headspace)
- Noom Alternative: Behavioral Telemetry Without Structural Disruption Modeling (/articles/disruption-modeling/noom)
- Spring Health Governs Human Care, Not Agent Coherence Loss (/articles/disruption-modeling/spring-health)
- Lyra Health vs Agent Coherence Governance: Two Different Diagnostic Objects (/articles/disruption-modeling/lyra-health)
- Ginger vs Agent Disruption Modeling: Behavioral Sensing for People, Structural Coherence for Agents (/articles/disruption-modeling/ginger-io)
- Cerebral vs Agent Disruption Modeling: Why Symptom-Driven Telepsychiatry and Structural Agent Coherence Are Different Problems (/articles/disruption-modeling/cerebral)
- Modern Health vs Structural Disruption Modeling for AI Agents (/articles/disruption-modeling/modern-health)
- Calm Business vs Agent-Level Disruption Modeling: A Different Layer of Coherence (/articles/disruption-modeling/calm-business)

- Anduril Counter-Drone Autonomy vs Agents That Diagnose Their Own Coherence Loss (/articles/disruption-modeling/anduril-counter-drone)
- Shield AI Hivemind vs. Disruption Modeling: external hardening or agent self-diagnosis? (/articles/disruption-modeling/shield-ai)
- Galileo OSNMA Authenticates the Signal, Not the Agent's Coherence (/articles/disruption-modeling/galileo-osnma)

Disruption Modeling overview → (/disruption-modeling)