Differential Interaction Testing (DIT)

A Black-Box Framework for Detecting Divergence in Outputs Consistent with Potential Composition Stability Composition Instability in AI Systems

Based on the Interaction Boundary Constraint (Harber, 2026)

The Problem

Every evaluation framework in production AI operates on a shared assumption: that evaluating a system's outputs is equivalent to evaluating the system. It is not.

Outputs cross the interaction boundary, the point where the participant encounters the result. The composition that produced the output (model state, component versions, routing paths, dependency configurations, intermediate transformations) is never present at that boundary. This is a structural condition, not a tooling gap. The Interaction Boundary Constraint (Harber, 2026) establishes that *evaluation beyond appearance requires composition to be present at the interaction boundary*. Where composition is not present, evaluation is limited to what is observable at the boundary.

Logging, tracing, and telemetry reconstruct composition after the fact. Reconstruction is not presence. Every reliability claim, consistency benchmark, and attribution judgment about AI systems today is grounded in inference over outputs, not in knowledge of what produced them.

The Reframe

Knowing the exact composition of a system at any given moment is for non-deterministic systems like LLMs, practically impossible. For deterministic systems, hashing all components gives a strong logical approximation but still operates at the logical layer, not the physical layer. The gap is structural.

But knowing what factors alter composition is an engineering problem. DIT does not attempt to see past the interaction boundary. It uses the boundary as the instrument. By submitting controlled probes and observing only what crosses the boundary (inputs and outputs), DIT detects composition instability without requiring access to internals, logs, or system architecture.

How It Works

DIT operates as an external harness. It requires one thing: an endpoint. The system under test does not know it is being evaluated. Four probe types target different dimensions of composition stability.

Probe Type	Method	What It Detects
Identity	Submit identical inputs at different times (minutes, hours, days, across deployment windows).	Temporal composition drift. If outputs diverge beyond baseline variance, composition changed between probes.
Isolation	Submit identical inputs varying one irrelevant variable (timestamp, IP, name, empty field, geographic origin).	Hidden sensitivities. If the output changes, composition depends on a variable it should not.
Sequence	Submit the same set of inputs in different orders.	Hidden state. If order affects output, the system carries state across interactions that influences composition.
Channel	Submit identical inputs from different entry points, origins, network paths, or client configurations.	Boundary inconsistency. If the observation channel itself behaves

		differently across entry points, DIT's own measurements may be affected.

Calibration

Non-deterministic systems produce different outputs for identical inputs by design. DIT must distinguish expected variance from composition-driven divergence.

Before probing begins, DIT runs a calibration phase. The same input is submitted multiple times in rapid succession, fast enough that composition is unlikely to have changed. This establishes the system's baseline output variance under stable composition.

Subsequent probes are evaluated against this baseline. The question is not "did the output change" but "did the output fall outside the baseline variance." This is a statistical test, not a binary comparison. The calibration phase must be repeated periodically to account for intentional changes to the system that shift the baseline.

Probe Design

The probes DIT runs determine what it can detect. Poorly chosen probes produce a clean stability map while real composition instability goes undetected. Probe selection is not incidental to the framework. It is the framework.

Identity Probe Frequency

Frequency should be tied to the system's change velocity. If deployments happen daily, probe daily. If dependencies update hourly, probe hourly. If the change velocity is unknown, that is itself a finding. Probe frequency should also span deployment windows, maintenance periods, and infrastructure rotation schedules.

Isolation Probe Variables

The variables tested should be derived from the system's specification. If the spec says geographic origin should not affect the output, test it. If the spec says time of day should not matter, test it. DIT tests the system's own claimed invariants against its actual behavior. Where no specification exists, start with: submission time, requester identity, network origin, input encoding, field ordering, and empty or null optional fields.

Sequence Probe Patterns

Start with the most common real-world interaction patterns and permute them. For agentic systems, this includes tool call sequences. For transactional systems, this includes common request flows. The goal is to detect hidden state that persists across interactions and influences composition.

Channel Probe Origins

Submit probes from different geographic regions, different client types, different authentication contexts, and different network paths. If the system treats DIT probes differently from real traffic due to routing or load balancing, channel probes expose this.

Coverage

For high-dimensional input spaces, exhaustive probing is impossible. DIT should operate in tiers. Tier one: probe the system's most common input patterns, derived from production traffic distributions. Tier two: probe known sensitive regions, inputs near decision boundaries, edge cases, and adversarial examples. Tier three: random sampling from the full input space as a coverage check. The stability map should report coverage explicitly, showing what portion of the input space has been probed and where the gaps are.

What DIT Produces

DIT produces a stability map: a time-indexed record of output consistency and divergence across probe types, conditions, and deployment windows. The map does not show composition. It shows what crossed the interaction boundary, when, under what conditions, and where output diverged beyond calibrated baseline variance. Composition change is inferred from that divergence, not observed directly. The stability map is evidence of boundary behavior. It is not a record of what produced it.

Each finding is specific: composition changed between these two timestamps, under these conditions, in response to this variable. This is sufficient to drive investigation, inform architecture decisions, and scope evaluation claims honestly.

A clean stability map means "no composition instability detected across these probes." It does not mean "composition is stable." The distinction matters. DIT coverage is bounded by probe diversity, and the stability map should always be read in the context of what was and was not probed.

Relationship to System-Internal Investigation

DIT detects from outside the boundary. Investigation happens behind it. These are complementary but epistemically different.

When DIT detects a divergence, the natural next step is to ask the engineering team what changed. They will check deployment logs, dependency updates, and configuration records. This relies on exactly the kind of system self-reporting that the Interaction Boundary Constraint identifies as structurally limited.

The paper does not pretend this step is unnecessary. It draws a distinction in epistemic status. DIT findings are boundary-external evidence that does not depend on the system reporting on itself. Log-based explanations are system-internal hypotheses about what changed. DIT provides the signal. Logs provide candidate explanations for the signal. The two operate at different levels of epistemic confidence, and reporting should reflect that difference.

Stability Thresholds

What counts as "stable enough" is a policy decision, not a measurement. DIT produces data. The organization interprets it.

Thresholds should be set based on risk tolerance, regulatory requirements, and the consequences of composition instability in the specific domain. A medical triage system might require zero unexplained divergences across all probe types. A content recommendation system might tolerate a defined rate of variance. A financial transaction system might set different thresholds for different transaction classes.

DIT provides the framework for setting and applying thresholds. It does not prescribe them. The stability map is the evidence. The threshold is the judgment.

Epistemic Position

DIT is still inference. The Interaction Boundary Constraint establishes that inference over outputs is structurally limited. DIT does not claim to escape this constraint.

DIT claims to occupy a stronger epistemic position than current evaluation methods. The field currently operates primarily on system self-report: logs, traces, telemetry, and explanations generated by the system being evaluated. DIT operates on boundary-external observation that does not require the system to report on itself.

The epistemic spectrum runs from system self-report (weakest) to boundary-external observation (stronger, where DIT operates) to composition present at the boundary (strongest, currently unachievable). DIT does

not claim to be at the top. It claims to be higher than where the field currently operates. That difference is meaningful even if it is not absolute.

Key Properties

Black-box by design. No access to code, logs, infrastructure, or model internals. The tool takes an endpoint and submits requests. This is the theoretical foundation, not a limitation.

System-agnostic. Works on rules engines, LLMs, agentic systems, MCP workflows, multi-server architectures, and distributed networks. Anything with an interaction boundary.

Non-cooperative. The system under test does not participate in its own evaluation. It does not report on itself. It does not know it is being probed.

Coverage-aware. The stability map reports what was probed, how thoroughly, and where gaps remain. Findings are always scoped to the probes that produced them.

Limitations

DIT can only detect composition changes that produce observable output differences for the specific inputs probed. Composition may change in ways that do not affect current outputs but would affect future outputs under different inputs. This is a blind spot that cannot be fully eliminated, only reduced through broader probe coverage.

DIT probes enter the system through the same boundary as real traffic. If the system treats DIT probes differently due to routing, load balancing, or other infrastructure decisions, the stability map describes the system's behavior under DIT probes, not under real usage. Channel probes partially address this but cannot fully resolve it.

For non-deterministic systems, the distinction between expected variance and composition-driven divergence depends on calibration quality. If calibration is performed during a period of composition instability, the baseline will be contaminated and subsequent measurements will be unreliable.

DIT does not explain why composition changed. It detects that it changed, when it changed, and what probe caught it. Diagnosis requires system-internal investigation, which operates at a lower epistemic level than DIT itself.

Theoretical Foundation

DIT is grounded in the Interaction Boundary Constraint (Harber, 2026), which establishes that across decisioning, agentic, multi-system (MCP), and network domains, evaluation can only reach what is observable at the interaction boundary. Composition is never present at that boundary. This applies independent of system design, correctness, or performance.

Rather than attempting to make composition visible, DIT systematically identifies what factors alter composition using only boundary-observable information. You do not need to see the wind to know it is blowing. You watch the trees.

DIT does not solve the interaction boundary constraint. It respects it.

DIT is proposed as an open-source framework. If you are building, evaluating, or regulating AI systems and want to contribute to a boundary-aware evaluation methodology, this is the starting point.