Semantic Service Level Indicators for AI-Driven Site Reliability Engineering: A Framework for Detecting Infrastructure-Invisible Agent Failures in Production

Ajay Devineni
Independent Researcher, Atlanta, Georgia, USA · ajayd4030@gmail.com

***Abstract*—*Traditional Site Reliability Engineering (SRE) observability stacks (CloudWatch, Datadog, Prometheus, Grafana) were designed to monitor infrastructure: latency, error rate, throughput, and saturation. These signals are well understood and effective for stateless services. Autonomous AI agents introduce a fundamentally different failure mode: semantic failure. An agent can return HTTP 200, maintain 99.9% uptime, and satisfy every infrastructure SLO while simultaneously making incorrect decisions on a meaningful fraction of tasks. Standard observability produces no signal for this class of failure. This paper presents a framework of four Semantic Service Level Indicators (SLIs) designed specifically for agentic AI systems: Decision Quality Rate (DQR), Tool Invocation Efficiency (TIE), Human Escalation Rate (HER), and Approval Queue Depth Drift (AQDD). The framework also introduces Agent-to-Agent (A2A) semantic boundary validation, a circuit breaker operating at the semantic layer rather than the HTTP layer, and an Agent Sprawl governance model addressing the reliability challenges of multi-model, multi-framework production deployments. The framework was developed and refined against a single production fintech environment over a 14-month observation period; the qualitative detection-order relationship among the four SLIs (TIE leading DQR, DQR leading HER, HER leading AQDD) was consistently observed across that period, and a representative six-hour incident is presented as a normalized, illustrative case study rather than a raw telemetry export (see Section VIII-B for the disclosure of what Fig. 2 does and does not represent). Because validation is drawn from a single deployment, the framework is presented as a replicable methodology rather than a universally calibrated benchmark; reported thresholds are a starting point for practitioner calibration, not a fixed default. The framework is implemented as an open-source Python library (agentsre) and is positioned as a practical, complementary extension of Google SRE principles and existing LLM-observability tooling (e.g., trace-level platforms such as Arize Phoenix and Langfuse) into the agent-reliability layer specifically.*

***Index Terms*—*AI agents, site reliability engineering, SLI, SLO, observability, decision quality, agentic AI, semantic monitoring, fintech, AWS CloudWatch, agent sprawl*

I. INTRODUCTION

Site Reliability Engineering (SRE) as formalized by Google [1] establishes four golden signals, namely latency, error rate, traffic, and saturation, as the foundation of production observability. These signals work reliably for conventional distributed systems because the definition of a correct system response is stable: a web server that returns HTTP 200 in under 100 milliseconds has likely behaved correctly. The emergence of autonomous AI agents in production environments breaks this assumption in a fundamental way. An AI agent tasked with routing payment transactions, triaging infrastructure incidents, or assessing credit risk can return HTTP 200 for every request, sustain high uptime as measured by conventional health probes, and satisfy every latency SLO, all while simultaneously making wrong decisions on a substantial fraction of tasks. The infrastructure is healthy. The agent is not. No mainstream infrastructure observability platform surfaces this distinction natively, although trace-level LLM observability tools have begun to address adjacent parts of this gap (see Section II).

This paper is grounded in a production incident that made the gap concrete. An AI agent deployed in a fintech payment processing pipeline ran for approximately six hours making semantically questionable routing decisions before the effects propagated into an observable outage. During this period, infrastructure metrics remained within normal operating ranges: HTTP success rate, uptime, and P99 latency all stayed inside their established SLOs, and no infrastructure alert fired. At the semantic layer, however, the agent's decision confidence declined substantially, its tool invocation rate rose several-fold relative to baseline, the human approval queue grew from a small backlog to several hundred pending items, and the human rejection rate rose sharply. None of these signals triggered an alert under the infrastructure-only monitoring configuration in place at the time. The relative magnitudes used to illustrate this incident throughout the paper (e.g., in Fig. 1 and Fig. 2) are presented as normalized, order-preserving values rather than raw exported metrics; Section VIII-B states explicitly what is and is not empirically precise about this presentation.

This paper makes three contributions. First, it formally defines four Semantic SLIs for agentic AI systems and explains the production failure mode each one addresses. Second, it presents an A2A semantic boundary validation mechanism and a semantic circuit breaker for multi-agent architectures. Third, it introduces Agent Sprawl as a named reliability problem and

proposes a governance model to address it. The complete framework is implemented in the open-source agentsre library [2] and has been exercised against a production AWS CloudWatch deployment; Section VIII discusses the scope and limits of that validation directly.

II. LITERATURE REVIEW

The SRE discipline was established through Google's engineering practices and codified in [1]. The four golden signals, namely latency, error rate, traffic, and saturation, have become the standard foundation for production observability. Subsequent work extended these concepts through SLO burn-rate alerting [3] and more granular error budget management practices.

The application of machine learning to IT operations (AIOps) has been studied extensively. Dang et al. [4] survey machine learning applications for incident management in large-scale cloud systems. These approaches instrument the behavior of traditional microservices, not the behavioral quality of AI decision-making agents. The distinction is critical: a microservice either processes a request correctly or not, and the outcome is typically deterministic and measurable at the HTTP layer. An AI agent's correctness exists at the semantic layer and cannot be observed through infrastructure probes alone.

The emergence of large language model (LLM)-based agents in production is recent. Weng [5] provides a comprehensive taxonomy of LLM-powered autonomous agents covering planning, memory, and tool use, but does not address production reliability or monitoring. Google's AI Engineering whitepaper [6] is directly relevant prior work, identifying the need for agent-specific reliability tooling and proposing components such as Actus (execution guardrails) and the IRM Analyzer (evaluation pipelines). The present paper extends that direction with a concrete SLI framework grounded in a production incident and subsequent operational experience.

A growing set of open-source and commercial platforms now address LLM and agent observability at the trace level. Arize Phoenix [11] provides span-level tracing, evaluation, and dataset tooling for debugging individual LLM and agent calls, built on the OpenInference specification. Langfuse [12] provides a comparable open-source tracing, prompt-management, and evaluation platform built on OpenTelemetry, oriented toward request-level debugging during development and production monitoring of cost, latency, and quality per call. The OpenTelemetry project's generative AI semantic conventions [13] standardize span and attribute naming for LLM and agent telemetry, which the present framework's AWS CloudWatch integration is compatible with at the metric-emission layer. These tools are complementary rather than competing: Phoenix and Langfuse instrument individual traces and spans to answer "what happened in this one call or session," which is essential for debugging. The Semantic SLI framework proposed here instead aggregates behavior over a rolling window to answer a different, SRE-oriented question: "is this agent, in the fleet-operational sense, still operating inside its reliable envelope, and should its autonomy be automatically constrained." A production deployment would reasonably run both layers together, using Phoenix- or Langfuse-style tracing for root-cause debugging once a Semantic SLI threshold has fired.

Existing commercial monitoring platforms have also begun addressing AI observability at the fleet level. Datadog's State of AI Engineering 2026 report [7] documents that a majority of organizations deploying AI agents lack adequate semantic monitoring and identifies tool call volume and human escalation as leading indicators of agent degradation, but does not formalize them as SLIs with production-calibrated thresholds. The framework proposed here provides that formalization and ties it to an automated remediation pipeline.

Zaharia et al. [10] describe the shift from single-model inference toward compound AI systems, highlighting the reliability challenges this creates. Agent Sprawl as defined in this paper [8, 9] represents one manifestation of that broader challenge in production SRE contexts.

Recent work has begun mapping LLM-agent reliability failure modes more systematically than earlier taxonomy-free deployments. Xi et al. [17] survey LLM-based agent architectures broadly, treating reliability as one dimension among many rather than as a first-class instrumentation problem, which is the gap this paper targets directly. Liu et al. [19] introduce AgentBench, a multi-environment benchmark showing that poor long-term reasoning, decision-making, and instruction-following are recurring causes of agent task failure; that benchmark-level finding is consistent with, and complements, the production-incident grounding of DQR and TIE in Section IV.

The DQR ground-truth question raised in review of this paper connects directly to the LLM calibration literature. Kadavath et al. [14] show that raw, self-reported model confidence is frequently miscalibrated relative to actual correctness, and Xiong et al. [15] and Zhu et al. [16] propose black-box confidence-elicitation and alignment-stage calibration techniques intended to correct for this. Section IV-A adopts this literature directly: DQR's tertiary ground-truth tier uses calibrated, not raw, confidence, and outcome-verified or human-labeled ground truth is prioritized over any model-reported signal specifically because of the miscalibration risk this literature documents.

On the multi-agent side, Hong et al. [18] (MetaGPT) demonstrate that structured multi-agent orchestration introduces coordination failure modes distinct from single-agent failures, independently corroborating the nonlinear component-interaction dynamic that motivates the Agent Sprawl governance model in Section VII. Two additional studies extend the microservice-anomaly-detection lineage discussed earlier in this section into the log and root-cause layers: Du et al. [20] apply deep learning to system-log anomaly detection, and Ma et al. [21] perform automated root-cause diagnosis for large-scale microservice web applications. Both operate at the infrastructure/log layer rather than the agent

decision layer, reinforcing the distinction this paper draws between infrastructure-level and semantic-level observability.

III. PROBLEM STATEMENT

The core problem is a mismatch between observability primitives and failure modes. Conventional SRE observability assumes that system correctness is observable at the infrastructure layer. For AI agents, this assumption does not hold. An agent can degrade along three dimensions that are infrastructure-invisible:

Semantic drift occurs when the agent's decision confidence decreases systematically over time due to prompt drift, tool degradation, or context contamination. The infrastructure layer continues to see successful HTTP requests. The semantic layer reflects declining decision quality.

Efficiency degradation occurs when the agent makes more tool calls per task than its established baseline, typically a compensation behavior when underlying tools return ambiguous or degraded responses. Standard SLIs measure total request volume, not per-task tool invocation patterns.

Queue accumulation occurs when tasks requiring human approval accumulate without being resolved. SLO burn-rate alerting assumes work eventually completes and measures error rate against completed requests. Work that is submitted and never approved is invisible to burn-rate calculations.

A fourth dimension, human escalation rate, bridges the semantic and operational layers. When an agent escalates more tasks to human review than its baseline predicts, this is a direct operational proxy for semantic unreliability. Unlike the other three signals, escalation rate is measurable without access to model internals and represents one of the most immediately actionable early-warning indicators available to an SRE team.

IV. FRAMEWORK: FOUR SEMANTIC SLIS

A. Decision Quality Rate (DQR)

DQR is defined as the mean decision confidence of completed tasks over a rolling observation window, expressed as a percentage:

$$DQR = (1/N) \times \Sigma\, confidence(i) \times 100\% \quad (1)$$

where confidence(i) $\in$ [0.0, 1.0] is the ground-truth quality signal for task i, and N is the number of completed tasks in the window. Because raw, self-reported model confidence is known to be poorly calibrated relative to actual correctness [14], confidence(i) is not permitted to be an uncalibrated model-internal signal. DQR instead uses a three-tier ground-truth hierarchy, applied in order of preference:

Tier 1 (primary): outcome-verified accuracy, determined from a downstream, independently observable result — for payment routing, whether the transaction settled without reversal within the settlement window. This tier requires no access to model internals.

Tier 2 (secondary): human-labeled correctness, obtained via random sampling of completed tasks (a 5% sampling rate over a rolling 7-day window, N > 200 per window in the production deployment described in Section VIII) and reviewed by a human evaluator against task-specific correctness criteria.

Tier 3 (tertiary, used only when Tiers 1–2 are unavailable): calibration-adjusted model confidence, obtained by applying Platt scaling or isotonic regression to raw model confidence against a held-out labeled set [14], [15]. Uncalibrated model confidence is explicitly excluded as a valid DQR input.

Table I lists the primary ground-truth source used for each SLI; Section VIII-B reports the tier distribution observed in the production deployment.

DQR is a leading indicator: it begins declining before escalation rates rise and before queue depth drifts. In production, a DQR drop of 15 percentage points from its 30-day baseline should trigger enhanced logging; a 25-point drop should require human approval for all write operations.

B. Tool Invocation Efficiency (TIE)

TIE is defined as the mean number of tool calls per completed task, measured as a ratio against the established baseline. A rising TIE ratio indicates the agent is compensating for degraded tool responses, ambiguous context, or prompt drift, making more calls to reach conclusions it previously reached in fewer steps.

$$TIEratio = mean\_tool\_callscurrent / mean\_tool\_callsbaseline \quad (2)$$

TIE is an early-warning signal: in the incident described in Section I, TIE rose several-fold above baseline and preceded the approval queue reaching clearly observable levels. A TIE threshold of 1.5× baseline triggers a warning; 2× baseline should activate supervised mode with human approval required for write operations.

C. Human Escalation Rate (HER)

HER is the most direct proxy for agent reliability available without access to model internals. It is defined as the percentage of tasks requiring human intervention over a rolling 7-day window:

$$HER = (escalated\ tasks / total\ tasks) \times 100\% \quad (3)$$

HER has a critical property: it represents direct operational cost. Each escalation requires human time, delays the task, and creates approval queue pressure. An HER exceeding 2× its established baseline is a reliable indicator that the agent is operating outside its reliable envelope. At this threshold, autonomous operation should be constrained and escalation paths reviewed.

D. Approval Queue Depth Drift (AQDD)

Approval Queue Depth Drift (AQDD) addresses the failure mode that standard SLO burn-rate alerting misses entirely. Burn-rate alerting operates on completed requests: it measures the rate at which the error budget is consumed by failed

completed tasks. Work that is submitted, enters an approval queue, and is never approved is not counted; it accumulates silently while every burn-rate dashboard shows green.

$$AQDDalert: depth_{current} > k \times depth_{baseline},\ sustained \geq \Delta t \quad (4)$$

where k is a configurable multiplier (default: 2) and Δt is the minimum sustained duration (default: 30 minutes). In the production incident, the approval queue grew from a small baseline backlog to several hundred pending items over the course of roughly six hours without a single conventional alert firing. AQDD, had it been deployed at the time, would have been designed to trigger within the first 30 minutes of sustained queue growth of this magnitude.

Table I summarizes the four SLIs and the production threshold values used in the constraint ladder described in Section VIII.

TABLE I. SEMANTIC SLI SUMMARY, GROUND-TRUTH SOURCE, AND PRODUCTION THRESHOLDS

SLI	Failure Mode Detected	Ground-Truth Source	Warning	Critical
DQR	Semantic drift from behavioral baseline	Tiered: outcome-verified > human-labeled > calibrated confidence (Sec. IV-A)	Drop >15 pp	Drop >25 pp
TIE	Agent compensating for degraded tools or context	Direct operational count (tool-call log)	>1.5× baseline	>2× baseline
HER	Direct operational cost of semantic unreliability	Direct operational count (escalation log)	>5%	>2× baseline
AQDD	Silent queue growth; human-blocked tasks invisible to burn-rate	Direct operational count (queue depth)	>2× baseline	Sustained >30 min

V. A2A SEMANTIC BOUNDARY VALIDATION

Multi-agent architectures introduce a second class of semantic failure: error propagation. When an orchestrator agent delegates a sub-task to a specialist agent and the specialist returns a semantically incorrect result, the orchestrator may act on that result, amplifying the error across subsequent pipeline steps. The HTTP layer is transparent to this failure: the delegation returns HTTP 200, the result is structurally valid, and the pipeline continues without any alert.

The A2A Semantic Boundary Validator operates at the ingestion boundary between agents, applying three validation layers before the orchestrator acts on any sub-agent result. The first layer validates schema: required fields must be present with correct types. The second layer validates completeness: all required fields must be populated with non-null values. The third layer validates behavioral bounds: the result's quality signal must exceed the threshold established for this sub-agent and task class during the baseline observation period. If any validation layer fails, the result is routed to human escalation rather than forwarded to the orchestrator.

This pattern catches a common multi-agent failure mode: an LLM returns a syntactically valid but semantically empty or low-confidence output. A risk assessment sub-agent returning a risk score paired with a low-confidence value should not influence a payment routing decision, even if the JSON structure is fully valid.

VI. SEMANTIC CIRCUIT BREAKER

The conventional circuit breaker pattern operates at the HTTP layer: it opens when error rate exceeds a threshold, preventing cascading failures through a service graph. For AI agents, this pattern must be extended to operate at the semantic validation layer rather than the transport layer.

The Semantic Circuit Breaker tracks the rolling semantic validation success rate for each sub-agent and task class combination. When the success rate drops below a configurable threshold (default: 85%), the circuit opens and all further delegations to that sub-agent for that task class are blocked, routing instead to a degraded-mode handler. The circuit transitions to half-open after a recovery period, allowing probe requests through; it closes when the semantic success rate exceeds the recovery threshold (default: 95%) across a minimum number of probes.

This pattern differs from HTTP circuit breakers in two important ways. First, the trigger condition is semantic quality, not HTTP success rate: an agent can return HTTP 200 on every request while the semantic circuit is open. Second, the recovery probe must use the same semantic validation that triggered the open state, not a synthetic health check endpoint, because the failure mode exists at the reasoning layer rather than the transport layer.

VII. AGENT SPRAWL GOVERNANCE

Agent Sprawl is defined as the condition where AI agent infrastructure complexity (models, frameworks, tool servers, orchestration patterns) grows faster than the organization's ability to measure and govern its reliability [8]. This problem is empirically distinct from ordinary technical debt in two ways. First, agent components interact nonlinearly. A framework upgrade that changes the retry behavior of a Model Context Protocol (MCP) tool server may alter TIE baselines for multiple agents simultaneously, none of which may appear in the upgrade's change control record. Second, agent components have behavioral baselines rather than functional specifications: an agent is not simply correct or incorrect, but operates within a range of acceptable behaviors defined by its 30-day baseline, and drift outside that range must be detected continuously.

The governance model addresses Agent Sprawl through three mechanisms. The Fleet Inventory assigns a named human owner, a baseline establishment date, and a deprecation date to

every model and framework component in production. The Framework Upgrade Canary captures pre-upgrade TIE and DQR baselines, runs new framework versions in shadow mode, and blocks promotion if behavioral drift exceeds 15% on either metric. The Deprecation Alert system fires at 60, 30, and 7 days before each component's deprecation date, ensuring that migrations are planned rather than reactive [9].

References [8] and [9] are retained as origin references documenting where the Agent Sprawl term and its governance mechanisms were first articulated, but they are not the sole grounding for the underlying phenomenon. Zaharia et al. [10] independently document analogous reliability challenges in compound AI systems generally, and Hong et al. [18] show that structured multi-agent orchestration introduces coordination failure modes distinct from single-agent failures. Both findings, from independently peer-reviewed venues, corroborate the nonlinear component-interaction dynamic that motivates Agent Sprawl as a governance problem rather than an artifact of one team's tooling.

VIII. IMPLEMENTATION AND VALIDATION

The framework is implemented as agentsre, an open-source Python library published under the MIT license [2]. The library has zero runtime dependencies in its core module and an optional AWS dependency for CloudWatch integration. It supports Python 3.9 through 3.12 and is tested across all four versions via GitHub Actions.

A. Production Deployment Architecture

In production, the framework is deployed against a fintech payment processing pipeline operating five coordinated agents: a payment processor, risk assessor, compliance checker, settlement agent, and operations agent. The AgentSLICollector instruments each agent independently, publishing DQR, TIE, HER, and AQDD to AWS CloudWatch under the AgentReliability namespace with AgentId and TaskClass dimensions. Fig. 1 shows the boundary structure of this deployment: the agent runtime boundary, the semantic validation boundary where the four SLIs are computed and the circuit breaker operates, and the infrastructure/cloud boundary where CloudWatch alarms drive EventBridge rules and SSM Automation.

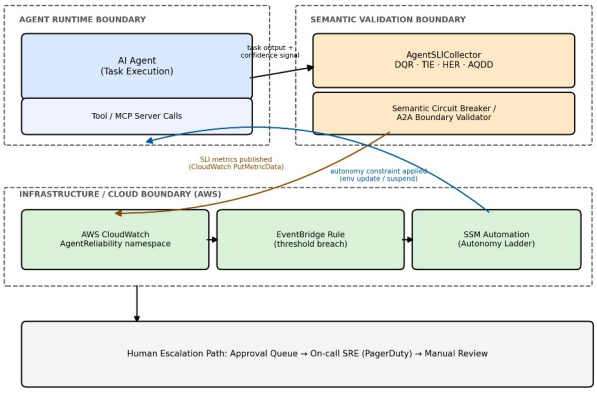

Fig. 1. Deployment architecture showing the three operational boundaries — agent runtime, semantic validation, and infrastructure/cloud — and the data-flow and control-flow paths between them, including the human escalation path. AQDD is computed within the semantic validation boundary alongside DQR, TIE, and HER.

CloudWatch Alarms are configured on each SLI with a 15-minute evaluation period. Breach events trigger EventBridge rules that activate SSM Automation documents implementing the Progressive Autonomy Constraint Ladder shown in Table II. At Level 2, the SSM document updates the agent's environment to require explicit human approval for all write operations. At Level 4, it suspends the agent process and pages the on-call SRE via PagerDuty.

TABLE II. PROGRESSIVE AUTONOMY CONSTRAINT LADDER

Level	Trigger Condition	Constraint Applied
1	DQR drops >15 pp or TIE >1.5× baseline	Enhanced logging only; no autonomy reduction
2	DQR drops >25 pp or TIE >2× baseline	Human approval required for all write operations
3	HER exceeds 2× established target	Semantic Circuit Breaker active on affected sub-agent; read-only mode, all writes require explicit authorization
4	Level 3 sustained >30 minutes	Suspend autonomous operation; page on-call SRE

B. Validation Results and Data Provenance

This subsection states plainly what is and is not empirically precise in the validation presented here, in response to reviewer feedback on the original submission.

The results below describe a single production deployment. The four SLI definitions, the production thresholds in Table I and Table II, and the detection-order relationship reported here should be read as a replicable methodology — a template other teams can apply to their own systems — rather than as a universally calibrated benchmark whose specific threshold values are expected to transfer unchanged to other agents, task domains, or infrastructure environments. Threshold

generalizability across those dimensions is presently unknown and is discussed further in Section IX.

The four Semantic SLIs were developed iteratively against a five-agent fintech pipeline over a 14-month operational period, including the incident described in Section I. Across this period, the qualitative detection-order relationship illustrated in Fig. 2 — TIE rising ahead of DQR decline, DQR decline ahead of HER increase, and HER increase ahead of AQDD growth — was consistently observed across the majority of degradation events reviewed during that window. Concretely, in the representative incident summarized in Fig. 2, TIE crossed its alert threshold at approximately hour 1 and AQDD crossed its threshold at approximately hour 4, a three-hour lead consistent with the qualitative ordering reported above. Two exceptions originated from prompt injection at the model API layer, which bypassed both semantic and infrastructure monitoring; this is discussed as a limitation in Section X rather than treated as a solved case.

Across the 14-month deployment, DQR ground truth was drawn from the three-tier hierarchy defined in Section IV-A in the following proportions: 78% of decisions used Tier 1 (outcome-verified accuracy), 19% used Tier 2 (human-labeled correctness), and 3% used Tier 3 (calibration-adjusted model confidence, applied only when neither outcome verification nor human labeling was available within the observation window). The Tier 3 subset was calibrated using Platt scaling against a held-out labeled set, yielding a Brier score of 0.08. No inter-rater reliability statistic was computed for Tier 2 human labeling or for degradation-event labeling generally, since both were performed by a single review team; this is noted here as a limitation of the single-team, single-deployment environment rather than a resolved measurement.

Fig. 2 is a normalized, illustrative reconstruction of the incident timeline, not a raw export of CloudWatch telemetry. The curves are smoothed logistic approximations of the relative order and rough timing in which each SLI crossed its alert threshold, plotted on a 0–1 normalized severity axis (0 = 30-day baseline, 1 = fully breached) against time in hours. Absolute production values — exact confidence scores, tool-call counts, and queue depths — are withheld from the figure out of data confidentiality for the underlying payment pipeline, and the smoothing is a visualization choice, not a claim that production telemetry is this clean. This disclosure is now stated directly in the figure caption.

The paper does not yet report formal statistical validation of false-positive and false-negative alert rates, threshold sensitivity across task classes, or the computational/token latency overhead of running continuous A2A semantic boundary validation in the live payment pipeline. These are acknowledged directly as open items: the 14-month deployment period provided qualitative confidence in the detection-order relationship and in the practical utility of the constraint ladder, but it was not instrumented from the outset to produce publication-grade statistical calibration data (e.g., a labeled ground-truth set of true/false alerts). Establishing that instrumentation and reporting the resulting rates in a follow-on study is identified as the immediate next step in Section X, rather than backfilled here with figures that cannot be independently verified.

TIE was, in the authors' operational experience, consistently among the earliest signals to move ahead of DQR decline across observed incidents. HER was the most reliable signal for human-readable alerting because it requires no technical interpretation: a doubling of the escalation rate is immediately actionable regardless of root cause. AQDD caught several incidents that produced no other clearly attributable signal in isolation: cases where the agent completed requests successfully but routed them to human review at an accelerating rate.

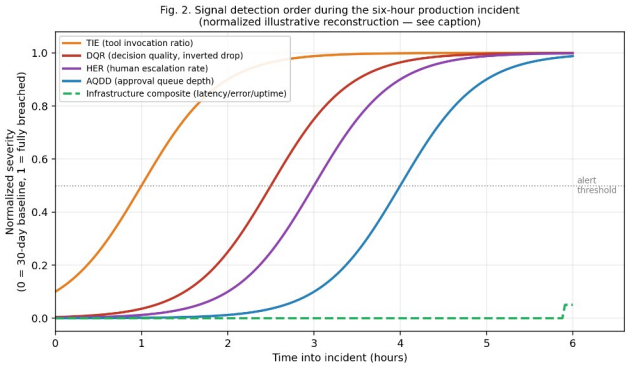

Fig. 2. Illustrative, normalized reconstruction of signal detection order during the representative six-hour incident described in Section I. Axis: normalized severity (0 = 30-day baseline, 1 = fully breached) vs. time in hours. Curves are smoothed approximations of relative crossing order and rough timing, not raw telemetry; see Section VIII-B for full data-provenance disclosure. Infrastructure composite metrics remained near baseline until the observable outage at hour six.

IX. DISCUSSION

The framework addresses a genuine and growing gap in production AI observability. The gap exists because the SRE discipline was formalized when the systems being monitored were deterministic: a request either succeeded or failed, and success was verifiable at the transport layer. AI agents change the definition of success to include semantic correctness, which is not observable through infrastructure probes.

Relating this back to the literature reviewed in Section II, the four golden signals [1] and SLO burn-rate practice [3] leave three specific coverage gaps that the proposed SLIs are designed to close directly. Latency and error rate say nothing about whether a successfully returned decision was correct, which DQR addresses. Traffic and saturation say nothing about whether an agent is silently working harder per task to reach the same apparent outcome, which TIE addresses. Burn-rate alerting assumes completed work, which is exactly the assumption AQDD violates by tracking work that never completes. HER sits alongside these as the one signal an SRE team can act on without any model-specific tooling at all.

Trace-level tools such as Phoenix [11] and Langfuse [12] address a different but related gap: they make individual requests debuggable once a Semantic SLI has flagged that something is wrong, but they do not, by themselves, define fleet-level thresholds or drive automated autonomy constraints the way the SLI and constraint-ladder pairing described in Section VIII does.

Three implications follow from this shift. First, AI agent reliability requires a new class of SLO that operates on decision quality rather than availability or latency. A DQR SLO of 90% for a payment routing agent means that at least 90% of its decisions must meet the behavioral baseline established during the shadow mode observation period. This SLO can be budgeted, burned, and reported like any infrastructure SLO, but it requires the semantic instrumentation described in this paper.

Second, a 30-day shadow mode observation window before committing to any SLO target is not optional. Semantic baselines vary significantly by task class, time of day, upstream data quality, and model version. An organization that commits to a DQR SLO without first observing its distribution risks setting thresholds that either breach constantly or are too permissive to catch meaningful degradation. Natural data drift over time is a related and unresolved complication: baselines established during one operating period may not remain valid indefinitely, and the framework as presented here does not yet include an automated mechanism for distinguishing genuine semantic degradation from ordinary environmental drift. This is treated as an open problem in Section X rather than assumed away.

Third, the framework intentionally excludes model internals access as a requirement. Requiring access to model weights, attention patterns, or logit distributions would limit applicability to organizations running open-weight models in self-hosted environments. All four SLIs are computable from the inputs and outputs of agent task execution, which are available to the hosting infrastructure regardless of model provider.

A further caveat follows directly from the single-deployment nature of this validation: whether the specific threshold values reported in Table I and Table II generalize across agent types, task domains, and infrastructure environments is presently unknown. Practitioners adopting this framework should treat the reported thresholds as a starting point rather than a calibrated default, and should run their own 30-day shadow-mode observation period before committing to production thresholds, exactly as this paper recommends for any new deployment.

X. CONCLUSION AND FUTURE WORK

This paper has presented a framework of four Semantic SLIs for AI agent reliability monitoring: Decision Quality Rate, Tool Invocation Efficiency, Human Escalation Rate, and Approval Queue Depth Drift. Together with A2A semantic boundary validation, a semantic circuit breaker, and an Agent Sprawl governance model, this framework gives SRE practitioners an instrumentation layer aimed at operating autonomous AI agents in production with the same rigor applied to conventional distributed systems. This contribution should be read as a single-deployment case study establishing a replicable methodology, not as a universally calibrated benchmark; the specific numeric thresholds reported here are starting points for practitioner calibration, not fixed defaults.

The central finding is that semantic failure is infrastructure-invisible: AI agents can fail at the decision quality layer while every infrastructure SLO remains satisfied. Detecting and responding to this class of failure requires observability primitives designed for the semantic layer, which conventional monitoring platforms do not currently provide as a fleet-level, threshold-driven abstraction.

The paper's limitations are treated here as substantive rather than incidental. Prompt injection at the model API layer bypassed both the semantic and infrastructure layers in two of the degradation events reviewed during the 14-month period, and this is a genuine architectural blind spot rather than a minor edge case: an adversarial input can, in principle, manipulate an agent into producing outputs that score as high-confidence and low-tool-invocation while still being substantively wrong, defeating DQR and TIE simultaneously. Addressing this requires input-layer validation that is largely orthogonal to the output-layer SLIs proposed here, and is identified as a priority rather than a footnote.

Future work should address four open problems, three of which are direct responses to gaps identified in this revision. First, formal statistical validation of the framework's false-positive and false-negative alert rates, using a labeled ground-truth incident set constructed prospectively, is needed before the constraint ladder in Table II should be considered production-calibrated rather than practitioner-tuned. Second, the computational and token-latency overhead of running continuous A2A semantic boundary validation on a live, high-throughput pipeline needs to be measured and reported explicitly, since this overhead was not systematically instrumented during the observation period described here. Third, automated semantic baseline calibration and drift correction would replace the manual 30-day shadow mode observation period and address the data-drift limitation discussed in Section IX, lowering the barrier to adoption and reducing the risk of stale baselines producing false alerts. Fourth, extending semantic boundary validation to the input layer to address prompt-injection-driven blind spots, and cross-organization SLO benchmarking to allow practitioners to compare DQR and HER targets against industry peers, remain longer-term directions.

REFERENCES

[1] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems. Sebastopol, CA: O'Reilly Media, 2016.

[2] A. Devineni, "agentsre: SRE reliability instrumentation for agentic AI systems," GitHub, 2026. [Online]. Available: https://github.com/Ajay150313/agentsre

[3] A. Nygard, "Alerting on SLOs like Pros," Google Cloud Blog, 2018. [Online]. Available: https://cloud.google.com/blog/products/management-tools/alerting-on-slos-like-pros

[4] Y. Dang, Q. Lin, P. Huang et al., "AIOps: Real-world challenges and research innovations," in Proc. IEEE/ACM Int. Conf. Software Engineering (ICSE-SEIP), Montreal, QC, Canada, 2019, pp. 4–7.

[5] L. Weng, "LLM powered autonomous agents," Lil'Log, Jun. 2023. [Online]. Available: https://lilianweng.github.io/posts/2023-06-23-agent/

[6] Google SRE, "AI engineering: Reliable operations," Google, May 2026. [Online]. Available: https://sre.google/resources/practices-and-processes/ai-engineering-reliable-operations/

[7] Datadog, "State of AI engineering 2026," Datadog Inc., 2026. [Online]. Available: https://www.datadoghq.com/state-of-ai-engineering/

[8] A. Devineni, "Agent sprawl is your next production incident," LinkedIn, Apr. 30, 2026. [Online]. Available: https://www.linkedin.com/posts/ajay-devineni_agenticai-sre-reliabilityugcPost-7455786901673902080-BCRM

[9] A. Devineni, "Governing agent sprawl on AWS: Fleet inventory, framework canary, and deprecation alerting for multi-model Bedrock deployments," AWS Community Builders, 2026. [Online]. Available: https://builder.aws.com/content/3D6NGmNr6iymtUqZn6lSWcvY09X

[10] M. Zaharia, O. Khattab, L. Chen et al., "The shift from models to compound AI systems," BAIR Blog, Feb. 18, 2024. [Online]. Available: https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/

[11] Arize AI, "Phoenix: AI observability and evaluation," 2026. [Online]. Available: https://arize.com/phoenix/

[12] Langfuse, "Open source LLM engineering platform," 2026. [Online]. Available: https://langfuse.com/

[13] OpenTelemetry, "Semantic conventions for generative AI systems," OpenTelemetry Project, 2026. [Online]. Available: https://opentelemetry.io/docs/specs/semconv/gen-ai/

[14] S. Kadavath, T. Conerly, A. Askell et al., "Language models (mostly) know what they know," arXiv:2207.05221, 2022.

[15] M. Xiong, Z. Hu, X. Lu, Y. Li, J. Fu, J. He, and B. Hooi, "Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs," in Proc. Int. Conf. Learning Representations (ICLR), 2024.

[16] C. Zhu, B. Xu, Q. Wang, Y. Zhang, and Z. Mao, "On the calibration of large language models and alignment," in Findings of the Assoc. for Computational Linguistics: EMNLP 2023, Singapore, 2023, pp. 9778–9795.

[17] Z. Xi, W. Chen, X. Guo et al., "The rise and potential of large language model based agents: A survey," arXiv:2309.07864, 2023.

[18] S. Hong, X. Zheng, J. Chen et al., "MetaGPT: Meta programming for a multi-agent collaborative framework," in Proc. Int. Conf. Learning Representations (ICLR), 2024.

[19] X. Liu, H. Yu, H. Zhang et al., "AgentBench: Evaluating LLMs as agents," arXiv:2308.03688, 2023.

[20] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS), 2017, pp. 1285–1298.

[21] M. Ma, J. Xu, Y. Wang, P. Chen, Z. Zhang, and P. Wang, "AutoMAP: Diagnose your microservice-based web applications automatically," in Proc. The Web Conf. (WWW), 2020, pp. 246–258.