Where Does the Error Live? A Three-Level Framework for Evaluating Multi-Agent Workflows in Production

Ahmed Elmokashfi
Amazon Web Services
rahelmok@amazon.com

Stefano Costanzo
Amazon Web Services
stefcos@amazon.com

Abstract

Agents that recommend under uncertainty—in medical triage, incident response, security operations—produce outputs whose correctness is revealed only when a human resolves the case. We present a three-level evaluation framework that exploits this structure: resolution records, generated as a byproduct of normal operations, serve as free ground truth requiring no labeling effort. Deployed on a large infrastructure diagnostic workflow, which comprises more than 90 steps and 70 individual agents, processing hundreds of incidents weekly, the framework operates at three granularities. Level 1 scores recommendations against resolutions and, firing on every case closure, turns quality into a time series that distinguishes interventions extending system reach from those sharpening specificity. Level 2 localizes failures within the agent graph via causal-graph alignment: an LLM-driven, anti-anchored procedure reconstructs the resolution's causal chain independently and traces misalignment to an originating agent, classifying errors into Knowledge, Reach, Attention, and Plumbing. Attention failures (evidence available but unused) form the majority of classified cases, suggesting that remediation should prioritize synthesis logic over knowledge or tooling—though we show this estimate carries only moderate inter-verifier agreement. Level 3 verifies reasoning trajectories against safety rules before the system acts; our central result there is a measurement, that LLM extraction non-determinism ($\pm$0.10 F1 across draws) is the dominant source of verifier instability, together with an architecture—a deterministic classifier that never extracts facts and so returns the same verdict on every run. Across all three levels, agreement degrades as the question becomes more localized (κ=0.631 on coarse verdicts to 0.431 on failure class), establishing that evaluation instruments require the same stability engineering as the systems they assess.

1 Introduction

AI agents are increasingly deployed as compositions—DAGs of sub-agents that parse, plan, execute, and synthesize in parallel. When such systems produce recommendations under uncertainty, evaluation faces a structural problem: ground truth arrives only after a human resolves the case, the output is a structured plan rather than a single answer, and a failure at one node may propagate silently through the graph. Existing benchmarks (GAIA [Mialon et al., 2023], AgentBench [Liu et al., 2024], SWE-bench [Jimenez et al., 2024]) sidestep these problems by treating agents as monolithic and providing pre-defined answers. Production systems have neither luxury.

We present a three-level framework that makes multi-agent evaluation actionable at each granularity, deployed on a large infrastructure diagnostic workflow, which comprises more than 90 steps and 70

40th Conference on Neural Information Processing Systems (NeurIPS 2026). Workshop: Who Verifies the Agents? Toward Reliable Agent Development.

individual agents, processing hundreds of incidents weekly. The framework applies wherever an agent recommends before an outcome is documented—medical triage (agent suggests workup; discharge summary reveals diagnosis), DevOps incident response (agent proposes root cause; postmortem documents what happened), security operations (agent classifies threat; analyst confirms true or false positive). In each case, the resolution record is a byproduct of normal operations that becomes retroactive ground truth at no labeling cost.

We decompose evaluation into three levels:

Level 1: End-to-End Scoring. Score each recommendation against the eventual resolution. Because evaluation fires on every resolution event, quality becomes a time series that distinguishes different intervention types without maintaining a curated benchmark.

Level 2: Per-Agent Attribution. When the score is low, localize the failure within the DAG via causal-graph alignment. An independently-reconstructed resolution graph is compared against the system's reasoning graph to identify the originating agent and failure category.

Level 3: Trajectory Verification. Before the system acts, check its reasoning against safety rules. LLM-based fact extraction makes the symbolic verifier's F1 swing by ± 0.10 between runs on identical input; a deterministic classifier that skips extraction detects flawed trajectories with identical results on every run.

Each level delivers actionable results. Level 1 separated two interventions that a single summary statistic would conflate. Level 2 localized the majority of failures to Attention (evidence available but unused)—pointing remediation toward synthesis prompts rather than knowledge or tooling. Level 3 provides a stable guardrail at 240ms per trajectory. Along the way, we characterize the reliability of each level: evaluation instruments are themselves non-deterministic, and their agreement degrades from κ=0.631 (coarse verdicts) to 0.431 (failure class) as the question becomes more localized.

Contributions. (1) A three-level framework deployed in production, demonstrating that compositional agent systems can be evaluated at multiple granularities using resolution records as free ground truth. (2) An LLM-driven, anti-anchored per-agent attribution procedure via causal-graph alignment over 254 executions, with a four-category failure taxonomy that routes each failure to a remediation owner, reported together with its inter-verifier reliability. (3) A deterministic trajectory verifier, and the measurement motivating it, that LLM extraction non-determinism—not the rules or the solver—is the dominant source of verifier instability. (4) Reliability characterization at each level, with validated negative results and design prescriptions.

2 Related Work

Agent evaluation benchmarks. GAIA Mialon et al. [2023], AgentBench Liu et al. [2024], SWE-bench Jimenez et al. [2024], and WebArena measure end-to-end task success without decomposing below the system level; OpenAI retired SWE-bench Verified in 2026 due to saturation OpenAI [2026]. Zhu et al. [2026] argue evaluation must account for component instantiation but survey no sub-system attribution method. Closest to our production setting, ITBench Jha et al. [2025] evaluates agents on real-world IT-automation tasks across SRE, FinOps, and CISO domains, but scores task completion end-to-end rather than localizing failures within a compositional agent graph. Our framework instead decomposes evaluation across three granularities, and we return to ITBench as the natural open-benchmark path to validating that decomposition beyond a single deployment (§7).

Scoring and judging. LLM-as-a-Judge Zheng et al. [2023] is standard, but Landesberg and Narayan [2026] show uncalibrated proxies can invert rankings, and Agent-as-a-Judge Zhuge et al. [2025] improves on it by inspecting intermediate artifacts. Guan et al. [2026] survey multi-turn evaluation but do not address compositional DAGs. We score against ground-truth resolutions rather than reference answers, and localize where a compositional failure originates rather than only whether it occurred.

Process reward models. The relative merits of outcome- versus process-level supervision for LLM reasoning remain debated Zheng et al. [2025]; AgentPRM Xi et al. [2025] and MASPRM Yazdani et al. [2025] extend step-level reward modeling to agents. Our Level 3 shares the step-level intuition

but separates deterministic detection from symbolic explanation, specifically to contain the extraction variance that reward-model approaches inherit.

Neuro-symbolic verification. FormalJudge Zhou et al. [2026] verifies facts via SMT solvers and NSVIF Su et al. [2025] formulates verification as constraint satisfaction. We extend this literature by quantifying the neural extraction layer's non-determinism (± 0.10 F1) and showing that a deterministic classifier, which never extracts facts, removes that run-to-run variation.

LLM non-determinism. Zatuchin [2026] decompose output noise via variance components and Yuan et al. [2025] show numerical/hardware sources perturb outputs even at temperature zero. We contribute the first measurement of this variance specific to trajectory verification.

Per-agent attribution. CAR Shah [2026] applies do-operations to agent causal models; CHIEF Wang et al. [2026] builds hierarchical causal graphs; CCPO Li et al. [2026b] and Li et al. [2026a] address credit assignment. Unlike these cooperative-reward settings, our agents have distinct heterogeneous roles (parser, enricher, planner, executor, synthesizer), and we localize failures by aligning against an independently-reconstructed resolution rather than by replaying a shared reward.

3 System Under Test

We evaluate a production diagnostic system for infrastructure operations. On ticket creation, it gathers telemetry, correlates signals, and posts analysis—typically before an engineer engages.

The system comprises over 70 agents in 7 waves: (1) parsing, (2) enrichment (KB, topology, impact), (3) planning per telemetry domain, (4) early posting, (5) execution against 100+ tools, (6) synthesis, (7) output formatting. Two tracks run in parallel: a Runbook track following codified procedures and a Gap-Analysis track exploring beyond them.

Three properties make evaluation hard. Agent prompts are thin ($\sim$20 lines); domain knowledge is fetched at runtime, so failures can be in knowledge sources rather than agent logic. Parallel tracks must be scored individually and in composition. Tool selection is itself evaluable: the right tool with wrong parameters fails differently from the wrong tool entirely.

4 Level 1: End-to-End Evaluation

4.1 Resolution-Triggered Scoring

Evaluation fires when a ticket is resolved. An evaluator agent retrieves the ticket history, locates the system's recommendation, and scores it against the engineer's resolution. Eligibility gates exclude tickets without substantive human comments or without a system recommendation; over the 14-week window these gates admit roughly one-third of resolved tickets, the remainder being auto-closed, duplicate, or lacking a documented diagnosis. The admitted set is therefore biased toward incidents that drew genuine human investigation—which is the population we care about for this measurement, but means absolute rates should not be read as system-wide coverage.

A note on ground-truth independence. Because the system posts before an engineer typically engages, a resolution could in principle anchor on the system's recommendation and inflate absolute scores. We mitigate this by scoring against the actions the engineer actually took and the final resolution state—not narrative text that could echo the system—and by gating out resolutions that restate the recommendation without corroborating action. The time-series analysis (§4.2) is further robust to a *constant* anchoring bias, since it reads changes across dated interventions rather than absolute levels. A fully anchoring-free measurement would require resolutions written without sight of the system output, which our deployment does not isolate; we flag this in §7.3.

Quality is scored 0–10 against the resolution (Table 1). When scores fall below 7, the evaluator emits structured gap analysis naming missing insights and tools—routing directly to remediation owners.

Score	Criteria
9–10	Correct root cause, specific mitigations matching resolution
7–8	Correct area, specific steps, minor gaps
5–6	Right direction, missing important details
3–4	Correct general area but vague or partially wrong
1–2	Minimally relevant
0	Wrong region/device/problem type

Table 1: Quality scoring rubric.

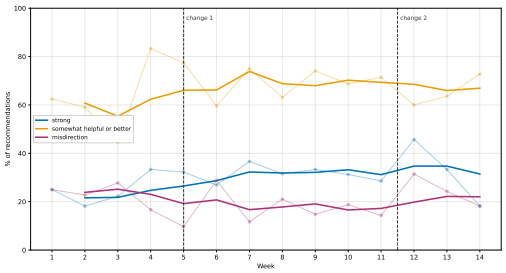


Figure 1: Weekly quality-score rates over 14 weeks (n=408), for the three bands tracked in the text: $\geq$8 (specific and actionable), $\geq$6 (directionally correct), and $\leq$4 (misdirection). Thin lines are per-week rates; bold lines are a 4-week rolling mean. Dashed verticals mark the two dated interventions discussed below (Change 1: extended tool reach; Change 2: domain grounding).

4.2 Quality as a Time Series

Figure 1 plots weekly rates over 14 weeks (n=408). We track three cuts: $\geq$8 (specific and actionable), $\geq$6 (directionally correct), and $\leq$4 (misdirection). Over the window these stand at 31.4%, 66.7%, and 20.3%.

Two interventions produce distinct, predictable signatures:

Change 1 (extended reach): Adding diagnostic tools lifts the floor—misdirection drops from 23.9% to 16.6% as previously-unobservable incidents move from the 3–4 band into the 5–6 band. The $\geq$6 rate plateaus in the high sixties: once tool coverage saturates common incident types, further reach has nothing to convert.

Change 2 (added specificity): Grounding in domain-specific data evacuates the middle of the distribution in *both* directions. The 5–7 band contracts by 14.4pp (p=0.022); mass moves to $\geq$8 (+6.6pp) and to $\leq$4 (+7.7pp). The mechanism: grounding makes recommendations more specific unconditionally, but the data covers the incident space unevenly—where it applies, hedged outputs become actionable; where it does not, previously-cautious outputs become confidently wrong. As the detailed results in Table 2 show, the $\geq$9 band is unchanged, confirming that grounding supplies specificity but not root-cause correctness. Two design lessons follow: (1) a single summary statistic would make these interventions indistinguishable; tracking the distribution separates them; (2) reach interventions compress the lower tail while specificity interventions widen the distribution at both ends.

5 Level 2: Per-Agent Attribution

End-to-end scoring identifies *that* the system failed; attribution identifies *where*. A flawed conclusion may originate in knowledge retrieval, planning, execution, or synthesis—and propagate through multiple agents before surfacing.

5.1 Method: Causal Graph Alignment

The procedure operates on one workflow execution and its resolved incident. It is executed by an *attribution agent*—an LLM prompted to follow the four phases below—not by human review or

Band	Before	After	Δ
Root cause identified (≥ 9)	17.0%	15.2%	-1.8
Specific and actionable (≥ 8)	30.1%	36.7%	$+6.6$
Middle (5–7)	51.1%	36.7%	-14.4
Misdirection (≤ 4)	18.8%	26.6%	$+7.7$

Table 2: Score redistribution across change 2 (n=329 before, n=79 after). The middle contracts significantly (p=0.022); mass moves to both tails.

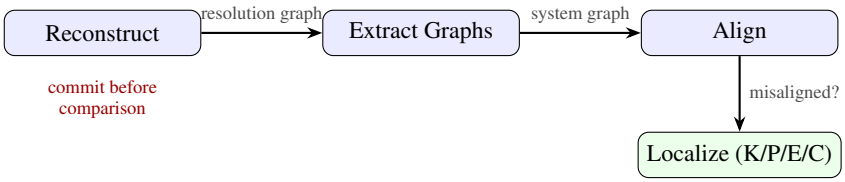


Figure 2: The four-phase attribution procedure, executed by an LLM agent. The agent reconstructs the resolution's causal chain and commits a verdict (Phase 1) before reading the system's recommendation; anti-anchoring prevents the judgement from inheriting the error it is meant to detect.

deterministic string matching; the only deterministic inputs are the system's own execution trace (agent outputs and their declared dependency edges) and the raw incident record. Its defining discipline is **commit-before-comparison**: the agent reconstructs the resolution and commits a verdict before it is allowed to read any prior evaluation or the system's own recommendation, so its judgement cannot anchor on the artifact it is judging.

Phase 1 (Reconstruct): From the incident record alone—the engineer's own correspondence, telemetry attached to the ticket, and the final resolution note, but *not* the system's output—the agent establishes the root cause, the decisive observation, and the path operators took to reach it. Cases with no documented diagnosis (auto-closed, duplicate, or deferred to a parent ticket) are marked *not assessable* and excluded, since there is no ground truth to align against.

Phase 2 (Extract): Build two graphs. The *resolution graph*: nodes are observations tagged by instrument and time; one node is marked decisive. The *system graph*: nodes are agent outputs; edges are declared dependencies (structural) and citation edges (which outputs the downstream agent actually used).

Phase 3 (Align): Match on *what was established*, not which tool established it. Key questions: is the decisive node present in the system graph? Does a citation path connect it to the conclusion? Do the conclusions assert the same causal direction?

Phase 4 (Localize): For a misaligned case, trace the decisive observation backward through the pipeline and stop at the *first* position where it was lost. Four positions are possible: **K** (Knowledge)—the mechanism was absent from the retrieved artifacts, so no downstream agent could have known it; **P** (Planning)—the knowledge was available but no planner requested the evidence; **E** (Execution)—a planner requested it but the executor failed to obtain it; **C** (Conclusion)—the evidence was obtained but synthesis dropped it or used it in the wrong causal role. Attribution targets this *originating* position, not the last agent to touch the output; the ordering $\mathrm{K} \to \mathrm{P} \to \mathrm{E} \to \mathrm{C}$ makes the choice deterministic once the loss point is identified.

From positions to classes. The four positions collapse into four classes by *remediation owner*. K is a *Knowledge* failure (fix the knowledge base). P or C is an *Attention* failure—reachable evidence was not requested (P) or not used (C)—owned by the agent prompts. E splits by cause: a missing *capability* (no tool can obtain the fact, or the input was unparsed) is a *Reach* failure owned by tooling, while a *fault* (error, timeout, skipped step) is a *Plumbing* failure owned by infrastructure. This is why a "failure in the knowledge source" (§3) is attributed to Knowledge, not to the retrieval agent: the taxonomy separates a silent knowledge base from an agent that failed to consult it.

Class	Count (share)	Implication
Attention (P/C)	117 (68%)	Fix prompts/synthesis
Reach (K/E)	39 (23%)	Add tools or fix parsing
Plumbing (E)	15 (9%)	Fix infrastructure
Knowledge (K)	1 (1%)	Expand knowledge base

Table 3: Failure-class distribution over the 172 misaligned cases (of 254 attributed; two-agent form). Attention dominates: evidence was available but not connected to conclusions. Shares carry the moderate failure-class agreement of Table 4, so we treat the class *ranking*, not the exact percentages, as the finding.

Question (increasing locality)	Raw agr.	κ	Landis-Koch
Alignment rating (5-way)	75.6%	0.631	substantial
Causal direction (binary)	69.3%	0.507	moderate
Failure class (K/R/A/P)	70.9%	0.431	moderate
Originating step (free text)	24.0%	—	unmeasurable

Table 4: Inter-verifier agreement between the two orchestration forms (n=254). We report raw agreement alongside κ because the failure-class distribution is highly imbalanced (Table 3), which deflates κ relative to raw agreement. Reliability degrades monotonically with locality.

5.2 Failure Distribution

Of the 254 attributed executions, 82 were rated aligned or not-assessable and 172 were misaligned; Table 3 gives the failure-class distribution over those 172, as judged by the two-agent (anti-anchored) form. The split is dominated by Attention, but the point estimates should be read against the moderate inter-verifier agreement on failure class reported in §5.3 (κ=0.431): the ranking of classes is stable, the exact shares are not.

The qualitative implication is robust to this uncertainty: Attention and Reach together account for the large majority of failures under either orchestration form, so the primary remediation levers are agent prompts and tooling—not the knowledge base, which originates at most a handful of failures. We did not perform human adjudication of the class labels; validating the exact split would require it, and we flag this in §7.3.

5.3 Attribution Stability

We deployed the methodology in two orchestration forms over all 254 attributed executions (single-agent vs. structurally anti-anchored two-agent workflow) and measured inter-verifier agreement:

A within-form test-retest (n=50) shows κ=0.786 for rating—the identical configuration still flips 7/50 ratings. Cross-form κ is lower (0.677), confirming orchestration adds disagreement above the sampling floor. The two-agent form is systematically harsher (36 tickets rated more severely vs. 11 more leniently, p=0.0003), suggesting anti-anchoring removes a leniency bias.

Disagreements are near-adjacent: of 62 rating disagreements, 47 are one-notch and zero are gross flips (aligned↔misaligned).

Implication. Attribution inherits agent non-determinism: the instrument that localizes failures is itself a non-deterministic agent whose reliability degrades with localization granularity. These numbers are lower bounds on instability—measured under near-ideal conditions (same model, same prompts, deterministic decoding). The practical consequence: coarse attribution (which failure class? which pipeline phase?) is reliable enough to direct remediation investment; fine-grained attribution (which exact step?) requires either a fixed taxonomy or human adjudication.

6 Level 3: Trajectory Verification

Levels 1 and 2 evaluate outcomes after the fact. Level 3 operates in-loop: before an agent acts on a high-stakes conclusion, is its reasoning sound?

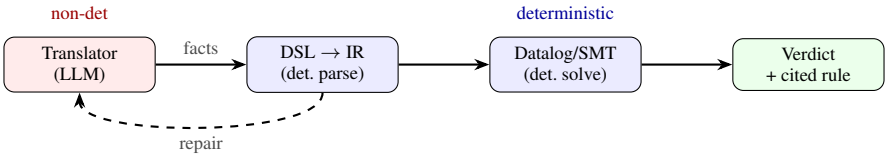


Figure 3: Verifier pipeline. Non-determinism is isolated at the Translator; everything downstream returns the same answer on the same facts.

Draw	Precision	Recall	F1
A	0.797	0.73	0.760
B	0.808	0.42	0.550

Table 5: Same input, same rules, same model, temperature 0. The $\pm$0.10 F1 swing lives entirely in recall—the extractor non-deterministically omits facts that rules depend on.

6.1 Mechanism

The verifier takes an agent's proposed action plus its reasoning chain and returns one of three verdicts: the action is consistent with the safety rules (entailment), violates one—with the rule named (contradiction)—or is unsupported (contingency). The procedure has three steps (see Fig. 3):

1. **Translate** (the one non-deterministic step): an LLM reads the trajectory and emits facts in a fixed vocabulary (`DisruptiveAction`, `CheckedCorrelationBefore`, etc.). A mechanical validator accepts or rejects each emission; rejected facts trigger bounded repair attempts.
2. **Evaluate** (deterministic): facts are tested against a rule library under closed-world semantics. Each rule is a conjunction that fires when all positive conditions hold and a required precondition fact is absent. A trajectory is flagged if any rule fires.
3. **Explain**: the fired rule's name and the missing precondition are returned. Because the verdict names *what was not checked* rather than merely refusing, the agent can self-correct—verify the missing condition and re-submit.

The architectural commitment is isolation: non-determinism is confined to step 1. Everything downstream is a fixed computation on the same facts. This design was deliberate; the finding below is that this single step is the entire reliability problem.

Rule provenance. Rules were mined from two sources: recurring patterns in 831 training-set violations, and 103 operational runbooks whose DO/DO-NOT constraints distilled into 11 candidate rules. Each passed a per-rule ablation: adopted only if it added true-error signal without disproportionately firing on clean trajectories. Several policy-backed rules were cut because the translator could not reliably emit the predicates they required—foreshadowing the finding below.

6.2 Central Finding: Extraction Non-Determinism

Holding everything fixed (712 trajectories, same rules, same model, temperature 0) and varying only the extraction draw (Table 5):

Precision holds ($\sim$0.80): when a rule fires, it is usually correct. The instability is in recall. Under closed-world semantics, a missing fact is treated as a violated precondition, so a single omitted `CheckedCorrelationBefore` flips the verdict. The problematic predicates are *judgement* predicates—whether some investigation "counts" as a correlation check—which are exactly the facts models least agree on. Three models on 154 trajectories agreed on only 22% of extracted facts (Jaccard 0.219); the largest model did not help.

6.3 Why the Obvious Fixes Fail

If the extractor is the problem, the natural move is to make it better. We tried four ways of doing so (Table 6), and none helped, for the same reason: each acts on the rules, the citations, or the model size, while the instability sits in the extractor's *judgement* about whether a fact holds.

Intervention	Effect on F1	Why it failed
Union of 3 extraction draws	−0.114	Spurious facts fire more rules
Require a cited source per fact	none	Citations are right; judgement is wrong
Ontology-normalized embeddings	none	Embedding already captures semantics
Larger model	−0.059	Larger is not more consistent

Table 6: Attempts to fix the extractor. None reduces extraction variance, which is the actual bottleneck.

		Variation	
Method	**F1**	**Run to run**	**Across CV folds**
Symbolic pipeline	0.550–0.760	±0.10	n/a (in-sample)
Embedding + LR	0.833	0	±0.010
Embedding + LR + structural	**0.853**	0	±0.005
Structural features only	0.729	0	—

Table 7: The symbolic pipeline's F1 changes from one run to the next on identical input; the classifier's does not. The two variation columns measure different things and should not be divided into one another: run-to-run variation is the instability that matters for a guardrail, while fold variation reflects sensitivity to the training split. Classifier results are 5-fold cross-validation over 7,023 unique prompts (38.5% flawed), deduplicated so that the clean and flawed versions of a scenario never fall in different folds.

The union result is the most telling. Pooling three draws should recover the facts a single draw drops, and it does—but it also pools each draw's spurious facts, so precision falls faster than recall rises. More extraction does not produce more reliable extraction.

6.4 The Fix: Detect Without Extracting

Since every fix that keeps the extractor in the decision fails, we take it out of the decision. The detector never converts the trajectory into facts. It embeds the whole trajectory with a fixed text-embedding model, adds six structural features computed directly from the action sequence (for example, `acted_before_investigating`, `disrupt_ratio`, `invest_ratio`, `tool_diversity`), and trains a logistic-regression classifier on the clean-versus-flawed label. Both the embedding and the features are deterministic, so the same trajectory always receives the same verdict.

Table 7 shows the effect. The classifier matches or exceeds the symbolic pipeline's best draw and removes run-to-run variation entirely. The structural features add 0.020 F1 and halve fold variation, and their coefficients make intuitive sense: acting before investigating predicts a flawed trajectory, and a higher share of investigative steps predicts a clean one. On their own, without any embedding or language model, they already reach F1 0.729.

6.5 Deployment: Screen Deterministically, Explain Selectively

The classifier tells us *whether* a trajectory is flawed but not *why*. The symbolic pipeline does explain why, by naming the violated rule, but it is unstable. The deployed verifier uses each for what it does well:

- *Screener (every trajectory).* The classifier makes the flag-or-pass decision: one cached embedding plus a logistic-regression evaluation, about 240 ms, with an identical result on every run.
- *Explainer (flagged trajectories only).* The symbolic pipeline runs only on what the screener flags, and returns the violated rule and the missing precondition. It explains about 74% of flagged true errors. Most of the rest are extraction misses; in roughly a third of them the extractor emits no facts at all, so no rule can fire.

We prefer the classifier for screening even though a zero-shot frontier-model judge is more accurate (F1 0.876–0.937 on a separate 120-trajectory audit sample; the two are not scored on the same set). The screen runs before every high-impact action, so it has to be fast, cheap, and consistent: an action

approved once should not be blocked when it is retried. The LLM judge takes seconds, costs a frontier-model call per check, and is itself non-deterministic. We pay that cost only where a human needs the explanation, in the explainer tier.

6.6 What the Result Does and Does Not Show

The 0.853 is not a production accuracy figure, for two reasons. First, the dataset is built in clean/flawed pairs, and the flawed versions were generated with a recognizable writing style: a TF-IDF model over the reasoning text alone reaches F1 0.986 by keying on hedging words that predict the flawed label about 99% of the time. Part of the classifier's score therefore reflects how the data was built rather than signal that would carry over to real trajectories; the structural features, which ignore the text, are the part most likely to transfer. Second, the classifier currently runs in shadow mode, and we make no claim about errors prevented in production.

What the level does establish is the measurement and the design that follows from it: LLM extraction is the dominant source of instability in this kind of verifier, and keeping it out of the detection decision is what makes the verifier stable.

7 Discussion

7.1 When Does This Framework Apply?

The three levels require different preconditions. Level 1 (end-to-end scoring) applies wherever three conditions hold: (1) an agent produces a structured recommendation before the outcome is known, (2) a human eventually documents the resolution, and (3) the resolution record is machine-readable. These conditions are met in medical triage (discharge summaries), DevOps (postmortems), security (analyst determinations), legal review (rulings), and customer support (closure reasons).

Level 2 (attribution) additionally requires that the agent system's internal execution trace be available—specifically, per-agent outputs and declared dependency edges. Systems that expose only a final output without intermediate artifacts cannot support graph alignment. Most workflow orchestration platforms (LangGraph, CrewAI, AWS AgentCore and custom DAG runners) retain this trace by default.

Level 3 (trajectory verification) requires that agent actions be expressible as a finite set of typed facts testable against domain-specific rules. This is natural for operational domains with codified safety procedures but less so for open-ended creative tasks.

7.2 Reliability Across Levels

A consistent pattern emerges from measuring the evaluation instruments themselves: stability degrades with localization granularity, and deterministic architecture is what restores it. Level 1 tracks trends and separates intervention types at the distribution level. Level 2 localizes failures to specific agents and classifies them into actionable categories; coarse verdicts are reliable (κ=0.631) and failure class is moderate (κ=0.431), both sufficient to direct remediation, while a within-form re-run still flips 14% of ratings. Level 3's extraction step swings $\pm$0.10 F1 across draws, so its screener gates actions with a deterministic classifier that returns identical verdicts on every run, and reserves the non-deterministic explainer for best-effort citation of the violated rule. The prescription is to use each level only for what its measured reliability supports.

7.3 Limitations

Ground-truth independence. The system posts before the engineer engages, so a resolution could anchor on its recommendation and inflate Level 1 scores. We score against the actions taken and final resolution state rather than echoed narrative, and the time-series reads changes rather than absolute levels (§4.2); but fully ruling out anchoring needs a cohort of resolutions written without sight of the system output, which our deployment does not isolate—the single most valuable next step.

Single domain, closed data. All results come from one infrastructure-diagnostics deployment with no released artifacts, so they are not independently reproducible and the transfer to other domains

is a conjecture about where the preconditions of §7 hold. A reproducible instantiation on a public benchmark such as ITBench Jha et al. [2025] is the clearest validation path; we leave it to future work.

Instrument validation. Level 1 uses a frontier-model judge, validated against human spot-checks for relative trends but without a reported agreement coefficient; its intervention comparisons are observational (before/after on dated changes, not controlled for incident-mix drift), so effect sizes are indicative. Level 2 measures inter-verifier *stability*, not *correctness*: the Attention majority rests on a moderate-agreement instrument (κ=0.431) and was not checked by human adjudication, so we treat the class ranking—not the exact shares—as the finding. Level 3's 0.853 is in-sample relative to the symbolic pipeline and carries a construction-artifact caveat; the contribution is the architecture and the non-determinism measurement, not a validated production detector.

Data handling. The evaluation set is production incident data processed within the operating organization's existing access controls, used only to evaluate the system that already handles these incidents, and not released; no personal data beyond operator identifiers already in the ticketing system is introduced by this work.

8 Conclusion

We presented a three-level framework that turns resolution records—a byproduct of normal operations—into continuous, labeled evaluation at no marginal cost. Each level yields what a monolithic score cannot: Level 1 separated reach from specificity interventions; Level 2 pointed remediation toward synthesis logic (the majority of localized failures were Attention) over knowledge expansion (which originated almost none); Level 3 gives a stable pre-action guardrail at 240 ms. The design needs only a machine-readable resolution record and a structured agent output, so it should transfer to other recommend-before-outcome domains—medical triage, incident response, security operations—though we tested one, and settling the transfer calls for an open-benchmark instantiation (e.g. ITBench Jha et al. [2025]).

Throughout, we present our reliability numbers—κ falling from 0.631 to 0.431 with locality, and $\pm$0.10 F1 extraction variance—as bounds on how each level should be used, not end-to-end guarantees: coarse attribution is reliable enough to prioritize investment, while exact failure-class shares and step-level localization await a fixed taxonomy and human adjudication.

References

Shengyue Guan, Jindong Wang, Jiang Bian, Bin Zhu, Jian guang Lou, and Haoyi Xiong. Evaluating LLM-based Agents for Multi-Turn Conversations: A Survey, 2026. URL `https://arxiv.org/abs/2503.22458`.

Saurabh Jha, Rohan Arora, Yuji Watanabe, Takumi Yanagawa, Yinfang Chen, Jackson Clark, Bhavya Bhavya, Mudit Verma, Harshit Kumar, Hirokuni Kitahara, Noah Zheutlin, Saki Takano, Divya Pathak, Felix George, Xinbo Wu, Bekir O. Turkkan, Gerard Vanloo, Michael Nidd, Ting Dai, Oishik Chatterjee, Pranjal Gupta, Suranjana Samanta, Pooja Aggarwal, Rong Lee, Pavankumar Murali, Jae wook Ahn, Debanjana Kar, Ameet Rahane, Carlos Fonseca, Amit Paradkar, Yu Deng, Pratibha Moogi, Prateeti Mohapatra, Naoki Abe, Chandrasekhar Narayanaswami, Tianyin Xu, Lav R. Varshney, Ruchi Mahindru, Anca Sailer, Laura Shwartz, Daby Sow, Nicholas C. M. Fuller, and Ruchir Puri. ITBench: Evaluating AI agents across diverse real-world IT automation tasks. 2025. URL `https://arxiv.org/abs/2502.05352`.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, 2024. URL `https://arxiv.org/abs/2310.06770`.

Eddie Landesberg and Manjari Narayan. Causal Judge Evaluation: Calibrated Surrogate Metrics for LLM Systems, 2026. URL `https://arxiv.org/abs/2512.11150`.

Mingxuan Li, Kai-Zhan Lee, and Elias Bareinboim. Counterfactual shapley credit assignment. *arXiv preprint arXiv:2607.16999*, 2026a.

Zhongyi Li, Wan Tian, Yikun Ban, Jinju Chen, Huiming Zhang, Yang Liu, and Fuzhen Zhuang. Counterfactual credit policy optimization for multi-agent collaboration. *arXiv preprint arXiv:2603.21563*, 2026b.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. *arXiv preprint arXiv:2311.12983*, 2023.

OpenAI. Why SWE-bench verified no longer measures frontier coding capabilities. urlhttps://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/, 2026. August 2026.

Jaineet Shah. Causal agent replay: Counterfactual attribution for LLM-agent failures. *arXiv preprint arXiv:2606.08275*, 2026.

Yiming Su, Kunzhao Xu, Yanjie Gao, Fan Yang, Cheng Li, Mao Yang, and Tianyin Xu. NSVIF: Neuro-symbolic verification on instruction following of LLMs. *arXiv preprint arXiv:2601.17789*, 2025.

Yawen Wang, Wenjie Wu, Junjie Wang, and Qing Wang. From flat logs to causal graphs: Hierarchical failure attribution for LLM-based multi-agent systems. *arXiv preprint arXiv:2602.23701*, 2026.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process Reward Models for LLM Agents via Step-Wise Promise and Progress, 2025. URL https://arxiv.org/abs/2511.08325.

Milad Yazdani, Mahdi Mostajabdaveh, Zirui Zhou, and Ying Xiong. MASPRM: Multi-agent system process reward model. *arXiv preprint arXiv:2510.24803*, 2025.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and Mitigating Numerical Sources of Nondeterminism in LLM Inference, 2025. URL https://arxiv.org/abs/2506.09501.

Dmitrij Zatuchin. Where Does the Noise Come From? A Variance-Components Decomposition of Non-Determinism in LLM Brand Answers. *arXiv preprint arXiv:2607.13304*, 2026.

Congmin Zheng, Jiacheng Zhu, Zhuoying Ou, Yuxiang Chen, Kangning Zhang, Rong Shan, Zeyu Zheng, Mengyue Yang, Jianghao Lin, Yong Yu, and Weinan Zhang. A survey of process reward models: From outcome signals to process supervisions for large language models. *arXiv preprint arXiv:2510.08049*, 2025. URL https://arxiv.org/abs/2510.08049.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Jiayi Zhou, Yang Sheng, Hantao Lou, Yaodong Yang, and Jie Fu. Formaljudge: A neuro-symbolic paradigm for agentic oversight, 2026. URL https://arxiv.org/abs/2602.11136.

Pengyu Zhu, Li Sun, Philip S. Yu, and Sen Su. The necessity of a unified framework for LLM-based agent evaluation. 2026. URL https://arxiv.org/abs/2602.03238.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Nn9POI9Ekt.