AgentEval: DAG-Structured Step-Level Evaluation for Agentic Workflows with Error Propagation Tracking

Dongxin Guo[1], **Jikun Wu**[2], **Siu Ming Yiu**[1]
[1]The University of Hong Kong [2]Stellaris AI Limited
bettyguo@connect.hku.hk, hk950014@connect.hku.hk,
smyiu@cs.hku.hk

Abstract

Agentic systems that chain reasoning, tool use, and synthesis into multi-step workflows are entering production, yet prevailing evaluation practices like end-to-end outcome checks and ad-hoc trace inspection systematically mask the intermediate failures that dominate real-world error budgets. We present AGENTEVAL, a framework that formalizes agent executions as evaluation directed acyclic graphs (DAGs), where each node carries typed quality metrics assessed by a calibrated LLM judge (GPT-4o), classified through a hierarchical failure taxonomy (3 levels, 21 subcategories), and linked to upstream dependencies for automated root cause attribution. An ablation study isolates the impact of DAG-based dependency modeling: it alone contributes +22 percentage points to failure detection recall and +34 pp to root cause accuracy over flat step-level evaluation with identical judges and rubrics.

Across three production workflows (450 test cases, two agent model families, predominantly sequential architectures with a 12% non-DAG trace rate), AGENTEVAL achieves 2.17× higher failure detection recall than end-to-end evaluation (0.89 vs. 0.41), Cohen's $\kappa = 0.84$ agreement with human experts, and 72% root cause accuracy against an 81% human ceiling. Cross-system evaluation on τ-bench and SWE-bench traces confirms transferability (failure detection recall ≥ 0.78) without taxonomy or rubric modification. A 4-month pilot with 18 engineers detected 23 pre-release regressions through CI/CD-integrated regression testing, reducing median root-cause identification time from 4.2 hours to 22 minutes and driving measurable failure rate reductions in two workflows.

1 Introduction

Agentic AI systems (autonomous agents that plan, reason, and execute multi-step workflows using external tools) are transforming NLP applications. As of early 2025, 62% of organizations are experimenting with AI agents and 23% are scaling deployments (McKinsey & Company, 2025), enabled by advances in agent frameworks (Wu et al., 2023; Schick et al., 2023; Patil et al., 2024). Yet evaluation infrastructure has not kept pace: current approaches rely on end-to-end outcome metrics that mask intermediate failures, ad hoc manual inspection that does not scale, or static benchmarks (Liu et al., 2024; Jimenez et al., 2024; Zhou et al., 2024b) disconnected from deployment constraints such as latency, cost, and continuous integration. This evaluation gap has tangible consequences: Gartner predicts that over 40% of agentic AI projects will be canceled by 2027, partly due to the inability to systematically evaluate deployed agents (Gartner, Inc., 2025).

Multi-step agent workflows create evaluation challenges absent from traditional NLP evaluation. Agent executions follow DAG-structured dependencies where errors propagate and compound through downstream steps (Cemri et al., 2025; Zhu et al., 2025), and workflows involve heterogeneous step types each requiring distinct quality metrics. Process supervision research demonstrates that intermediate-step assessment outperforms outcome-only evaluation in the RL training setting where per-step ground truth is available (Lightman et al., 2024; Uesato et al., 2022); we adapt this insight to inference-time assessment, where ground truth is constructed from expert annotation rather than learned from rewards, and where deployment infrastructure must operate continuously rather than as a one-time training signal.

We present AGENTEVAL,[1] an evaluation infrastructure addressing these challenges through four contributions:

1. **Evaluation DAG formalization**: representing

[1]The name "AgentEval" may have been used by other unrelated projects; this work is independent and identified by its GitHub repository (URL at the end of Section 7).

agent workflows as DAGs where each node carries typed quality metrics, enabling step-level evaluation with error propagation tracking.

2. **Step-level quality metrics**: a metric suite evaluated via calibrated LLM-as-judge scoring (GPT-4o) with demonstrated human alignment ($\kappa = 0.84$).
3. **Hierarchical failure taxonomy**: a three-level taxonomy with 21 subcategories derived from 523 agent traces, with quantitative error propagation statistics.
4. **Automated regression suite**: evaluation infrastructure integrated with CI/CD pipelines for continuous quality monitoring.

On three production workflows with predominantly sequential architectures (using Claude 3.5 Sonnet and Llama 3 70B as agents with GPT-4o as an independent judge), AGENTEVAL achieves 2.17× higher failure detection recall than end-to-end evaluation, $\kappa = 0.84$ human agreement, and 72% root cause accuracy. We explicitly characterize the approach's boundaries for more dynamic architectures in §5.5.

2 Related Work

Agent Evaluation and Process Supervision. Agent evaluation benchmarks have proliferated (Liu et al., 2024; Jimenez et al., 2024; Zhou et al., 2024b; Yao et al., 2024; Ma et al., 2024), with LATS (Zhou et al., 2024a) providing per-step value estimates for planning. However, a recent survey (Yehudai et al., 2025) identifies a critical gap: existing benchmarks evaluate agent *capabilities* in controlled settings, whereas deployment requires evaluation *infrastructure* with continuous monitoring, regression detection, and CI/CD integration. The process supervision literature demonstrates that evaluating intermediate steps outperforms outcome-only evaluation (Lightman et al., 2024; Uesato et al., 2022), while Cemri et al. (2025) and Zhu et al. (2025) show that error propagation is the primary bottleneck in agent performance. The LLM-as-judge paradigm (Liu et al., 2023; Zheng et al., 2023) enables scalable evaluation with >80% human agreement.

Production Infrastructure. The ML observability ecosystem, including MLflow, Weights & Biases Weave (Weights & Biases, 2024), LangSmith (LangChain, Inc., 2023), Arize Phoenix (Arize AI, Inc., 2023), Braintrust (Braintrust Data, Inc., 2023), Inspect (AI Security Institute, UK, 2024), and AgentOps (AgentOps, Inc., 2024), provides monitoring for ML pipelines and emerging agent support. These tools differ from AGENTEVAL in one important way: none provides formal DAG-based dependency modeling with error propagation tracking and root cause attribution (Table 7). AGENTEVAL's root cause attribution draws motivation from software engineering fault localization (Jones and Harrold, 2005; Abreu et al., 2007), operating as a practical greedy heuristic rather than formal causal inference.

3 The AGENTEVAL Framework

AGENTEVAL comprises four integrated components: (1) a formal DAG representation for agent workflows, (2) step-level quality metrics evaluated via calibrated LLM-as-judge, (3) a hierarchical failure taxonomy, and (4) an automated regression suite. Figure 1 provides an architectural overview.

3.1 Agent Workflow as Evaluation DAG

We formalize agent workflows as evaluation DAGs to enable structured step-level assessment.

Definition 1 (Evaluation DAG). An evaluation DAG is a tuple $\mathcal{G} = (V, E, \tau, \mathcal{M})$ where $V = \{v_1, \ldots, v_n\}$ is a set of evaluation nodes corresponding to agent steps; $E \subseteq V \times V$ defines directed dependency edges; $\tau : V \to \mathcal{T}$ maps each node to a step type from $\mathcal{T} = \{\text{PLAN}, \text{TOOLSEL}, \text{PARAMGEN}, \text{EXEC}, \text{SYNTH}\}$; and $\mathcal{M} : V \to 2^{\mathbb{M}}$ maps each node to applicable quality metrics based on its type.

Each node v_i carries input context c_i from parent nodes $\text{pa}(v_i)$, agent output o_i, and optional reference r_i. The step-level quality score is $q(v_i) = \text{Eval}(o_i, r_i, c_i, \mathcal{M}(v_i))$. The five-type classification is extensible: the framework supports adding types (e.g., MEMRETRIEVAL for RAG agents) by defining new metric sets and rubrics.

The DAG structure enables independent step evaluation, error propagation tracking, counterfactual analysis, and parallel evaluation of independent branches. AGENTEVAL supports both *schema-defined DAGs* (expected workflow structure) and *trace-inferred DAGs* (actual execution path), enabling detection of structural deviations such as missing steps or unexpected branches. Non-DAG traces (∼12%) are handled via loop unrolling and timestamp-based branch resolution, with 0.8% falling back to flat evaluation (Appendix A). During our pilot, structural deviation between schema-

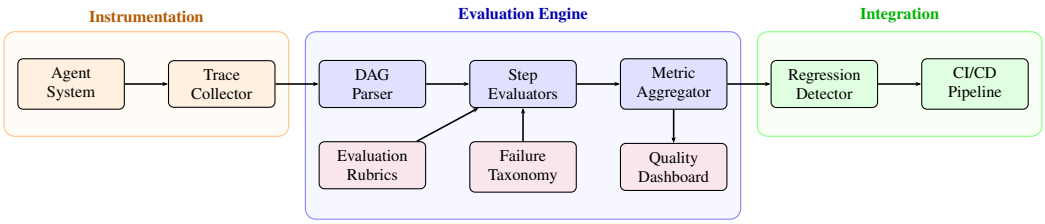

Figure 1: AGENTEVAL architecture. Agent traces are collected via OpenTelemetry-compatible instrumentation, parsed into evaluation DAGs, assessed by type-specific step evaluators using calibrated LLM-as-judge rubrics and the failure taxonomy, aggregated into workflow-level metrics, and fed into regression detection and CI/CD integration.

defined and trace-inferred DAGs proved a useful quality signal: deviant traces were associated with 2.1× higher failure rates than conformant traces. This is an observational pattern, not a causal claim; an agent may legitimately deviate from its expected DAG when the environment returns unexpected output, and deviation alone does not imply malfunction.

3.2 Step-Level Quality Metrics

Each step type has dedicated metrics: PLAN steps are assessed for completeness and feasibility; TOOLSEL for selection accuracy and relevance; PARAMGEN for parameter correctness and completeness; EXEC for success rate and result validity; and SYNTH for faithfulness, completeness, and coherence. We employ LLM-as-judge evaluation using **GPT-4o** (`gpt-4o-2024-08-06`) with temperature $= 0$. The agent LLMs (Claude 3.5 Sonnet and Llama 3 70B) are from *different model families* than the judge, addressing circular bias. Each metric uses a structured prompt with a 1–5 rubric, chain-of-thought reasoning, and 5 few-shot calibration anchors per metric (Appendix C). By "calibration" we mean stratified few-shot anchoring with 5 examples spanning the score range, not iterative prompt optimization against a held-out human-labeled set.

Prompt design: absolute versus relative framing. The judge prompts use two intentionally different framings depending on step type. PLAN is scored relative to the original user query, since planning is the step that translates the query into workflow structure and has no meaningful upstream context (Appendix C, C.2). Subsequent steps such as TOOLSEL are scored relative to the local context they received from upstream nodes rather than re-evaluated against the original query (Appendix C, C.1). As a consequence of this asymmetry, a downstream step that handles flawed upstream input gracefully (e.g., a tool call that retries on a malformed parameter) will not be flagged as a local failure, while a step that compounds the error will. Downstream prompts include prior step outputs but not prior judge scores (Algorithm 1, line 4), so judges do not bias toward agreement with upstream attributions.

3.3 Hierarchical Failure Taxonomy

We derive a three-level failure taxonomy from manual analysis of 523 agent traces **entirely disjoint** from the 450 evaluation test cases (Table 1). The taxonomy is informed by prior work on multi-agent failures (Cemri et al., 2025; Zhu et al., 2025). The 9 Level 2 categories are used for detection and primary classification; 21 Level 3 subcategories (Appendix B) provide granular diagnostic detail. Taxonomy classification runs as a separate LLM-judge call after a step has already been flagged as a failure by score thresholding; it produces a diagnostic label and does not interact with the propagation rule in Algorithm 1. Removing the taxonomy (Table 4) therefore degrades RCA primarily through loss of semantic grouping during error analysis, not through changes to which steps are attributed as root cause.

3.4 Automated Regression Suite

The regression suite enables continuous quality monitoring through: test case specification as JSON with expected DAG structures and tolerance thresholds, versioned evaluation tagged with model/prompt/tool configurations, regression detection via paired bootstrap tests ($p < 0.05$) with dual-threshold alerting (historical 2σ *and* bootstrap significance), CI/CD integration via GitHub Actions that blocks deployment on critical regressions, and progressive evaluation where fast smoke tests (10 cases, <5 minutes) gate full suites (100+ cases,

Level 1	Level 2	Freq.
Planning	Goal misinterpretation	12%
	Missing steps	8%
	Incorrect ordering	5%
Execution	Wrong tool selection	18%
	Parameter errors	22%
	API/tool failures	9%
Integration	Context loss	11%
	Output hallucination	10%
	Premature termination	5%

Table 1: Hierarchical failure taxonomy derived from 523 agent traces (disjoint from evaluation data). Context loss has the highest downstream amplification factor (3.2×).

Algorithm 1: AGENTEVAL Evaluation Procedure

Input: Agent execution trace T, DAG schema S (optional)
Output: Evaluation report R with step scores, failures, root causes

$\mathcal{G} \leftarrow \text{ParseDAG}(T, S)$; `// Reconstruct DAG`
$\langle v_1, \ldots, v_n \rangle \leftarrow \text{TopologicalSort}(\mathcal{G})$;
for v_i ***in*** $\langle v_1, \ldots, v_n \rangle$ **do**
  $c_i \leftarrow \text{AggregateContext}(\text{pa}(v_i))$;
  $q(v_i) \leftarrow \text{LLMJudge}(o_i, r_i, c_i, \mathcal{M}(\tau(v_i)))$;
  if $q(v_i) < \theta_{\tau(v_i)}$ **then**
    $f_i \leftarrow \text{ClassifyFailure}(v_i, \text{Taxonomy})$;
    `// Greedy heuristic: select lowest-scoring parent`
    if $\exists\, v_j \in pa(v_i) : q(v_j) < \theta_{\tau(v_j)}$ **then**
      $v_j^* \leftarrow \arg\min_{v_j \in \text{pa}(v_i)} q(v_j)$;
      Mark v_i as *propagated* from v_j^*;
    else
      Mark v_i as *root cause*;
    end
  end
end
$R \leftarrow \text{Aggregate}(\mathcal{G}, \{q(v_i)\}, \{f_i\})$;
return R;

<1 hour), reducing cost by 80% during development.

Algorithm 1 details the core evaluation procedure. For each trace, the DAG is reconstructed, nodes are processed in topological order, and each step is evaluated with type-specific metrics via the LLM judge. Failures are classified using the taxonomy and attributed as either root-cause or propagated using a greedy heuristic: when multiple parents have low scores, the parent with the lowest quality score is selected as the propagation source. This prioritizes simplicity and low false attribution rates over formal causal reasoning (Appendix F).

Specification	CS	DA	DP
Agent LLM	Claude 3.5 Sonnet		Llama 3 70B
Framework	LangChain		Custom
# Tools available	8	6	5
Avg. steps/trace	6.1	5.2	3.8
# Test cases	150	150	150
Edge case %	22%	18%	15%

Table 2: Workflow specifications. Agent LLMs are from different model families than the GPT-4o judge.

4 Experimental Setup

4.1 Agent Workflows

We evaluate on three agent workflows deployed in customer service (CS), data analysis (DA), and document processing (DP) pipelines (Table 2). Each uses the same tool set and prompt templates as its production counterpart but operates on anonymized historical queries.

4.2 Evaluation Dataset

We use two **completely disjoint** datasets: a *taxonomy development set* of 523 traces (January–March 2025), and an *evaluation dataset* of 150 test cases per workflow (450 total, April–June 2025), sampled using stratified random sampling (∼80% typical, ∼20% edge cases). Ground truth annotations were produced by 5 domain experts (3+ years experience), *none of whom are paper authors*. Each test case is a multi-step trace, and annotation is performed at the step level: the human-evaluation subset of 150 traces yields 987 step annotations in total, each rated 1–5 on the type-specific rubric. Of these, 195 steps are rated ≤ 2 by the human majority and constitute the failing-step set used as the Failure Detection Recall denominator (Section 4.2; full step distribution in Appendix W). For human evaluation, 9 separate domain experts independently rated 150 cases on a 1–5 rubric (inter-annotator $\kappa = 0.79$, Fleiss' $\kappa = 0.76$). Root causes were independently annotated with inter-annotator $\kappa_{\text{RCA}} = 0.74$ (81% pairwise agreement), which we treat as the human ceiling.

4.3 Baselines and Metrics

End-to-End Only (E2E): evaluates only final output quality using GPT-4o with no intermediate assessment. **Flat Step Evaluation (Flat)**: evaluates each step independently using identical GPT-4o judge and rubrics but without DAG structure. **Rule-Based**: 47 hand-crafted deterministic rules across the three workflows.

Method	FDRec↑	FPR↓	HA (κ)↑	RCA↑
E2E Only	.41 [.35,.47]	.08 [.05,.12]	.52 [.45,.58]	N/A
Flat Step	.67 [.61,.73]	.15 [.11,.20]	.71 [.65,.76]	.38 [.30,.46]
Rule-Based	.58 [.52,.64]	**.05** [.03,.08]	.63 [.57,.69]	.45 [.37,.53]
AGENTEVAL	**.89** [.84,.93]	.07 [.04,.11]	**.84** [.79,.88]	**.72** [.64,.79]

Table 3: Main results on 150 human-annotated test cases with 95% bootstrap CIs. AgentEval vs. Flat Step: $p < 0.001$ (FDRec, HA, RCA).

We measure **Failure Detection Recall (FDRec)**, **False Positive Rate (FPR)**, **Human Agreement (HA, Cohen's κ)**, and **Root Cause Accuracy (RCA)**. All metrics reported with 95% bootstrap CIs (10,000 resamples).

5 Results and Analysis

5.1 Main Results

AGENTEVAL achieves FDRec 0.89, representing $2.17\times$ higher recall than E2E (0.41) and outperforming Flat Step (0.67, +22 pp, $p < 0.001$). FDRec is computed against the 195 steps humans labeled as failing; for each method, the question is which of those 195 failing steps the method flags. By construction, E2E only inspects the final synthesis step, so failures originating earlier in the workflow are detectable only when they cause the final output to fail visibly. The $2.17\times$ gap therefore quantifies the recall lost to that structural restriction, not a difference in judge quality on equivalent inputs. Since the only difference from Flat Step is DAG structure (identical judges and rubrics), the +22 pp improvement over Flat Step further isolates the contribution of dependency modeling on top of step-level evaluation. Across all 450 test cases, 63% of step-level failures are *propagated* from upstream errors rather than locally caused, underscoring why dependency tracking is essential. Human agreement reaches $\kappa = 0.84$ [0.79, 0.88] ("almost perfect"), remaining strong on the failure subset ($\kappa = 0.76$, Appendix D). Root cause accuracy of 72% approaches the human ceiling (∼81%), with 72% of incorrect attributions within 1 DAG hop of the true root cause (Appendix E). Results are consistent across all three workflows (Appendix H).

The DAG advantage holds on both typical (+23 pp FDRec) and edge cases (+19 pp), confirming the benefit is not driven by edge case oversampling (Appendix G). The advantage grows with workflow length: +15 pp FDRec for ≤3 steps vs. +28 pp for ≥6 steps. On non-DAG traces (∼12%), performance degrades moderately (FDRec 0.82, RCA 0.58) but remains above flat evaluation.

Configuration	FDRec↑	HA (κ)↑	RCA↑
Full AGENTEVAL	**.89** [.84,.93]	**.84** [.79,.88]	**.72** [.64,.79]
− DAG structure	.67 [.61,.73]	.71 [.65,.76]	.38 [.30,.46]
− LLM judge (replaced w/ rules)	.62 [.56,.68]	.66 [.60,.72]	.51 [.43,.59]
− Failure taxonomy	.82 [.77,.87]	.79 [.74,.84]	.54 [.46,.62]
− Calibration	.85 [.80,.90]	.76 [.70,.81]	.68 [.60,.75]

Table 4: Ablation study. DAG structure contributes the most (−22 pp FDRec, −34 pp RCA). All differences significant at $p < 0.01$.

5.2 Ablation Study

Removing DAG structure causes the largest degradation (−22 pp FDRec, −34 pp RCA; $p < 0.001$), confirming dependency modeling as the most important design decision. Replacing LLM-as-judge with rules causes the second-largest drop (−27 pp FDRec). Removing the failure taxonomy primarily impacts RCA (−18 pp), and calibration anchors primarily affect human agreement (−8 pp κ).

Execution failures dominate across all workflows (42–55%), but integration failures are most impactful: context loss has an average downstream amplification factor of $3.2\times$, explaining why DAG-based evaluation outperforms flat evaluation. Results are robust across four judge models (GPT-4o, Claude 3.5 Sonnet, GPT-4o-mini, Llama 3 70B), with the DAG advantage remaining significant for all ($p < 0.01$; Appendix I). Same-family judging (Claude judging Claude agents) produces comparable or slightly *lower* scores, ruling out self-evaluation inflation (Appendix J).

Judge capability requirements. The judge does not need to solve the agent's task in order to evaluate it: verifying that a tool selection is reasonable, that parameters match a schema, or that a synthesis step is grounded in retrieved context is in general easier than producing the action de novo. This asymmetry is what keeps the cost profile tractable, since a cheaper-than-agent judge can still rate steps usefully. Empirically, weaker judges preserve the DAG advantage in failure detection but degrade more sharply on root cause accuracy: GPT-4o-mini retains FDRec 0.83 against 0.89 for GPT-4o, but RCA drops from 0.72 to 0.65 (Appendix I). The practical implication is that the judge should be at least comparable to the weakest agent in the workflow on relevant verification tasks; deployments

using judges substantially weaker than the agent should expect detection to remain useful while attribution becomes less reliable.

Counterfactual Validation. To validate root cause identification, we replace identified root causes with gold-reference outputs and re-evaluate downstream steps. Across 30 sampled failing traces, counterfactual correction confirms AGENTEVAL-identified root causes in 87% of cases (26/30), with downstream scores improving by an average of 2.3 points. This procedure measures the combined effect of correcting the upstream output and the judge's sensitivity to the improved context that downstream steps then receive; it does not isolate either component. The engineering implication is that fixing the attributed root cause typically yields measurable improvement on downstream steps the agent itself would re-execute, which is precisely the intervention pattern engineers use during debugging. Cases where counterfactual correction does not improve downstream scores ($\sim$13%) are flagged for manual review rather than auto-attributed.

5.3 Regression Detection

We simulate six model update scenarios across 18 scenario–workflow combinations. AGENTEVAL achieves 88% precision and 94% recall for regression detection, correctly identifying no regression for benign prompt rephrasing and correctly localizing degradation for single-step tool deprecation. E2E evaluation requires 4$\times$ more test cases for equivalent statistical power. During the pilot, 23 regressions were detected organically (8 genuine, 12 borderline, 3 false positives). One illustrative case: a prompt template update for CS-Agent replaced a structured instruction with a free-form variant; AGENTEVAL flagged a significant drop in PARAMGEN quality for the account-lookup step ($p < 0.01$, FDRec dropped 12 pp), which propagated to downstream steps, and the team reverted within hours. Additional case studies appear in Appendix L.

5.4 Cross-System Generalization

To assess generalization, we evaluate on τ-bench (Yao et al., 2024) (120 traces) and SWE-bench (Jimenez et al., 2024) (80 traces) without modifying the taxonomy or rubrics. AGENTEVAL successfully parses 90% of τ-bench and 89% of SWE-bench traces into valid DAGs. FDRec remains robust ($\geq$0.78), while RCA degrades more sharply (-14 pp τ-bench, -20 pp SWE-bench), concentrated in novel failure patterns not in our taxonomy. To assess potential annotator bias, 2 independent annotators (graduate students, not authors) re-annotated 40 τ-bench traces with comparable agreement ($\kappa = 0.74$ step quality), suggesting limited confirmation bias. This pattern of robust detection with degraded attribution on novel domains suggests the DAG-based approach generalizes as evaluation structure, while the taxonomy benefits from domain-specific extension.

Setting	Domain	FDRec	RCA
Internal	CS-Agent	.92	.75
	DA-Agent	.88	.78
	DP-Agent	.86	.64
τ-bench	Retail	.83 [.76,.89]	.61 [.52,.70]
	Airline	.79 [.71,.86]	.55 [.45,.64]
SWE-bench	Code editing	.78 [.69,.86]	.52 [.42,.62]

Table 5: Cross-system generalization. Performance degrades on external benchmarks but remains above E2E baselines (FDRec 0.38 on τ-bench, 0.35 on SWE-bench).

5.5 Architecture Scope Analysis

Our internal agents exhibit a 12% non-DAG rate. Simulation of increasing non-DAG rates (Appendix K) shows that the DAG advantage remains significant up to $\sim$60%, beyond which it becomes marginal (<5 pp). The boundary is not a property of the architecture alone but of how much execution structure can be reconstructed post hoc: retry loops are recoverable through unrolling, dynamic branches through timestamp resolution, and short reasoning loops through bounded iteration limits. Architectures that defeat reconstruction tend to share two properties: long unbounded reasoning loops, and parallel agent handoffs without explicit dependency declaration. AGENTEVAL is therefore most useful for the sequential and moderately-branching tool-calling patterns that currently dominate production deployments; we view extension to cycle-aware and multi-agent settings as the natural next step (Section 7).

6 Deployment Experience

AGENTEVAL operates as a sidecar service with OpenTelemetry-compatible trace collection (<2% latency overhead). Evaluation runs asynchronously with a tiered judge fallback: GPT-4o primary, GPT-4o-mini secondary, and locally deployed

Metric	Value	Source
Total traces evaluated	12,847	System logs
Unique evaluation runs	342	System logs
Pre-release regressions detected	23	Alerts + triage
Genuine / Borderline / False	8 / 12 / 3	Team triage
Config. changes from findings	14	Git commits
Failure rate reduction (CS)	31%→18%	Eval suite
Failure rate reduction (DA)	27%→15%	Eval suite
Root-cause time (before)	4.2 hr (med.)	Survey (n=47)
Root-cause time (with AE)	22 min (med.)	Logs (n=156)

Table 6: Pilot deployment outcomes (4 months, 18 engineers). Note: baseline and pilot measurements use different instruments; the magnitude of improvement should be interpreted with caution (details in Appendix M).

Feature	AE	LangSmith	Arize	Braintr.	Inspect
DAG dependency model	✓	∼	✗	✗	✗
Error propagation	✓	✗	✗	✗	✗
Root cause analysis	✓	✗	✗	✗	✗
Calibrated LLM judge	✓	∼	∼	∼	✓
Failure taxonomy	✓	✗	✗	✗	✗
CI/CD regression	✓	✓	∼	✓	∼
Open-source	Avail.	✓	✓	✓	✓

Table 7: Comparison with evaluation tools (public documentation, January 2026; features evolving rapidly). ✓: full; ∼: partial; ✗: not supported.

Llama 3 70B as an API-independent fallback. Step-level evaluation takes ∼2 seconds per step; cost per trace is ∼$0.02 with GPT-4o-mini. Multi-judge aggregation (3× cost) is recommended for release-gating only. At scale (100K traces/day), daily cost is ≈$2K with GPT-4o-mini, motivating progressive evaluation where fast smoke tests gate full suites.

Over a 4-month pilot (September 2025–January 2026), three engineering teams used AGENTEVAL to evaluate agents during active development (Table 6). The most impactful outcome was the replacement of a previously manual root-cause investigation process with an automated propagation-aware analysis: median time to identify a root cause moved from 4.2 hours under the prior workflow (self-reported survey) to 22 minutes with AGENTEVAL (system-logged). The two figures use different measurement instruments and the precise magnitude should be read with that caveat (Appendix M); the directional improvement is consistent across all three teams. AgentEval's error propagation analysis revealed that 23% of CS-Agent failures originated from a context truncation bug in the policy retrieval step; this single fix reduced CS-Agent's failure rate by 8 percentage points. For DA-Agent, taxonomy-guided analysis identified "parameter hallucination" (syntactically valid but semantically incorrect SQL WHERE clauses) as 31% of execution failures; targeted few-shot examples reduced the parameter error rate from 27% to 11%. Practitioner feedback highlighted error propagation visibility as the primary benefit: "The DAG visualization made error propagation chains immediately visible. We found three cascading failure patterns that end-to-end testing had missed entirely." (Senior Engineer). Key deployment lessons include the importance of taxonomy-first design and principled calibration anchor selection (Appendix N). Onboarding a new workflow requires ∼20–30 person-hours, with same-domain workflows requiring less (∼12–18 hrs) through partial reuse.

7 Conclusion

We presented AGENTEVAL, an evaluation infrastructure that formalizes agent workflows as evaluation DAGs with step-level quality metrics, a hierarchical failure taxonomy, and automated regression testing. On three production workflows with predominantly sequential architectures, AGENTEVAL achieves 2.17× higher failure detection recall than end-to-end evaluation, $\kappa = 0.84$ human agreement, and 72% root cause accuracy approaching the human ceiling (81%). The ablation study confirms DAG-based dependency modeling as the single most impactful component, an advantage that grows with workflow complexity and holds across four judge models. Cross-system evaluation on τ-bench and SWE-bench demonstrates transferability (FDRec ≥ 0.78), while architecture scope analysis characterizes the practical boundary at ∼60% non-DAG traces. A 4-month pilot validated practical utility, with specific AgentEval-informed fixes driving measurable failure rate reductions. Future work includes extending to multi-agent systems via cycle-aware evaluation graphs, automated taxonomy evolution through trace clustering, and integration of evaluation signals into agent training loops.

Reproducibility. All materials are available at: https://github.com/bettyguo/AgentEval.

Acknowledgments

We thank The University of Hong Kong and Stellaris AI Limited for their support of this work, and the anonymous reviewers for constructive feedback that improved the paper.

Limitations

Our evaluation covers predominantly sequential agent architectures; the DAG advantage diminishes beyond ∼60% non-DAG trace rates, limiting applicability to highly dynamic multi-agent systems (§5.5). All core results are from a single organization, though cross-system evaluation on two external benchmarks provides evidence of generalizability. LLM-as-judge evaluation introduces model-dependent biases; our cross-family design mitigates but does not eliminate these. The RCA algorithm is a practical heuristic, not formal causal inference. The deployment comparison uses different measurement instruments and should be interpreted with caution. We evaluate English-language workflows only. The commercial tool comparison is time-sensitive. Extended discussion of all limitations appears in Appendix O.

Ethical Considerations

Our evaluation uses anonymized historical queries with no personally identifiable information; data handling was reviewed under our organization's internal data governance process. Human annotators were domain experts compensated at standard professional rates and informed of the study's purpose. LLM-as-judge evaluation introduces potential biases from the judge model (GPT-4o); we mitigate this through cross-family evaluation design but acknowledge residual model-dependent biases. The failure taxonomy was developed on internal data and may not capture failure modes relevant to all deployment contexts.

References

Rui Abreu, Peter Zoeteweij, and Arjan J.C. van Gemund. 2007. On the accuracy of spectrum-based fault localization. In *Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION (TAICPART-MUTATION 2007)*, page 89–98. IEEE.

AgentOps, Inc. 2024. AgentOps: Observability and DevTool platform for AI agents. https://github.com/AgentOps-AI/agentops.

AI Security Institute, UK. 2024. Inspect ai: Framework for large language model evaluations. *Zenodo*.

Arize AI, Inc. 2023. Arize Phoenix: AI observability and evaluation. https://github.com/Arize-ai/phoenix.

Braintrust Data, Inc. 2023. Braintrust: The AI observability platform for building quality AI products. https://www.braintrust.dev/.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya G. Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Why do multi-agent LLM systems fail? *arXiv preprint*, arXiv.2503.13657.

Gartner, Inc. 2025. Gartner predicts over 40% of agentic AI projects will be canceled by end of 2027. Gartner Press Release, June 25, 2025. https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

James A. Jones and Mary Jean Harrold. 2005. Empirical evaluation of the tarantula automatic fault-localization technique. In *20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005), November 7-11, 2005, Long Beach, CA, USA*, pages 273–282. ACM.

LangChain, Inc. 2023. LangSmith: AI agent and LLM observability platform. https://www.langchain.com/langsmith.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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, and 3 others. 2024. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval:

NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 2511–2522. Association for Computational Linguistics.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

McKinsey & Company. 2025. The state of AI in 2025: Agents, innovation, and transformation. McKinsey Global Survey. https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *arXiv preprint*, arXiv.2211.14275.

Weights & Biases. 2024. W&B Weave: A toolkit for developing AI-powered applications. https://github.com/wandb/weave.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *arXiv preprint*, arXiv.2308.08155.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint*, arXiv.2406.12045.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. 2025. Survey on evaluation of llm-based agents. *arXiv preprint*, arXiv.2503.16416.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024a. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 62138–62160. PMLR / OpenReview.net.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024b. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. 2025. Where LLM agents fail and how they can learn from failures. *arXiv preprint*, arXiv.2509.25370.

A Non-DAG Trace Handling

Approximately 12% of traces involve patterns that do not conform to strict DAGs, primarily retry loops (8%) and dynamic branching (4%). Retry loops are handled by *unrolling* into sequential attempts within the DAG, preserving the most recent attempt's output as the step's canonical result. Dynamic branches are resolved using trace timestamps to reconstruct the executed path. For traces where DAG reconstruction fails entirely (0.8%), the system falls back to flat step evaluation and flags the trace for manual review.

B Full Failure Taxonomy

Table 8 presents the complete three-level failure taxonomy with Level 3 subcategories.

Construction methodology. The taxonomy was developed through affinity diagramming. Three authors independently coded the same 523 development traces (entirely disjoint from the 450 evaluation test cases), generating an initial pool of failure descriptors. These descriptors were then merged

into candidate categories through multiple consensus sessions. The resulting 9 Level 2 categories and 21 Level 3 subcategories were locked before any evaluation data was annotated; the evaluation set was used only to confirm that the categories were comprehensive enough to label observed failures, never to refine the taxonomy itself. During the pilot deployment, three additional subcategories were proposed and merged through a propose-discuss-merge governance protocol (Appendix N); these additions are reflected in the version reported here.

C LLM-as-Judge Prompt Templates

All prompts use GPT-4o (`gpt-4o-2024-08-06`) with temperature $= 0$ and `max_tokens=1024`.

C.1 Tool Selection (TOOLSEL)

System: You are an expert evaluator assessing the quality of an AI agent's tool selection in a multi-step workflow. Evaluate strictly according to the rubric below. First provide step-by-step reasoning, then your score on a separate line as "Score: N".
Task: Given the agent's current context and the tool it selected, rate the tool selection quality on a 1–5 scale.
Rubric:
5: *Optimal*. Best available tool for the subtask.
4: *Acceptable*. Valid tool but better alternative exists.
3: *Partially correct*. Correct category but suboptimal tool.
2: *Poor*. Wrong category but may produce some useful output.
1: *Incorrect*. Cannot accomplish the subtask.
Calibration Examples: [5 anchor examples omitted for space]
Context: {agent_context}
Available tools: {tool_list}
Selected tool: {agent_tool_selection}
Subtask description: {subtask}

C.2 Planning (PLAN)

System: You are an expert evaluator assessing the quality of an AI agent's execution plan. First provide reasoning, then "Score: N".
Task: Rate the plan quality on a 1–5 scale given the user query and available tools.
Rubric:
5: *Complete & optimal*. All subtasks identified, feasible, correctly ordered.
4: *Mostly complete*. Minor omission but critical steps present.
3: *Partial*. Covers main task but misses 1–2 important subtasks.
2: *Inadequate*. Major gaps or infeasible steps.
1: *Incorrect*. Plan would not address the user query.
Calibration Examples: [omitted for space]
User query: {query}
Available tools: {tool_list}
Agent plan: {agent_plan}

D Human Agreement by Score Subset

Subset	n (steps)	κ
All steps	987	.84
Failing (score ≤ 2)	195	.76
Borderline (score $= 3$)	146	.69
Passing (score ≥ 4)	646	.87

Table 9: Human agreement by score subset. Agreement remains substantial on failures ($\kappa \geq 0.69$), confirming the overall $\kappa = 0.84$ is not inflated by majority-class agreement.

E RCA Error Analysis

Among the 28% of incorrect attributions, three patterns dominate: (1) *multi-branch convergence* (41% of RCA errors), where independent upstream failures converge at a single step; (2) *long propagation chains* (33%), where the true root cause is $\geq$3 steps upstream; and (3) *threshold boundary cases* (26%), where the root cause scores marginally above the threshold.

The median distance between attributed and true root cause for incorrect attributions is 1 hop, with 72% of errors within 1 hop and 91% within 2 hops. This means most errors send engineers to a *nearby* step rather than a distant one, substantially reducing misattribution cost compared to random assignment (mean distance 2.8 hops).

F RCA Attribution Strategy Comparison

Attribution Strategy	RCA Acc.	False Attr.
Greedy lowest-parent (default)	.72	.06
Full-path minimum	.75	.14
Weighted propagation	.73	.09

Table 10: Comparison of RCA strategies. Full-path minimum improves accuracy by 3 pp but doubles false attributions to distant ancestors.

Tradeoff discussion. The three strategies differ in how aggressively they search upstream for a root cause when multiple parents have low scores. Greedy lowest-parent stops at the immediate parent with the lowest quality score. Full-path minimum walks the entire ancestor chain and selects the globally lowest-scoring node. Weighted propagation interpolates between the two by weighting parent scores by edge confidence.

Full-path minimum achieves a 3 pp RCA improvement, but at the cost of more than doubling

Level 1	Level 2	Level 3	Description & Example
Planning	Goal misinterpretation	Scope error	Agent addresses wrong aspect of the query
		Ambiguity failure	Agent selects wrong interpretation without clarification
	Missing steps	Tool omission	Required tool not included in plan
		Verification gap	Plan lacks result verification
		Prerequisite skip	Dependency step omitted
	Incorrect ordering	Dependency violation	Step executed before prerequisites
		Suboptimal sequence	Valid but inefficient ordering
Execution	Wrong tool selection	Category error	Wrong tool category
		Granularity mismatch	Correct category but wrong specificity
	Parameter errors	Type mismatch	Wrong data type
		Value error	Correct type but incorrect value
		Missing required	Required parameter omitted
	API/tool failures	Timeout	Tool call exceeds time limit
		Error response	Tool returns error, agent fails to handle
Integration	Context loss	Truncation	Information dropped between steps
		Selective omission	Agent ignores relevant output parts
	Output hallucination	Fabrication	Information not grounded in any result
		Conflation	Information from different steps incorrectly merged
	Premature termination	Partial completion	Agent declares success prematurely
		Loop exit	Agent exits retry loop too early

Table 8: Complete three-level failure taxonomy with 21 Level 3 subcategories.

the false-attribution rate (.06 to .14) by reaching for distant ancestors that may have been transient or already compensated for by intermediate steps. During pilot use, engineers reported that distant misattributions were disruptive: the wasted investigation time on a misattribution three hops upstream consistently exceeded the time saved by the 3 pp accuracy gain on cases where a distant attribution was correct. We therefore selected greedy lowest-parent as the default. Weighted propagation offers a smaller intermediate tradeoff (1 pp RCA improvement, +3 pp false attribution) and may suit deployments with reliable edge-confidence signals, though our pilot did not require it.

G Stratified Results

Subset	Method	FDRec	HA (κ)	RCA
Typical ($\sim$80%)	Flat Step	.64	.70	.36
	AGENTEVAL	**.87** [.81,.92]	**.83**	**.70** [.61,.78]
Edge ($\sim$20%)	Flat Step	.76	.74	.44
	AGENTEVAL	**.95** [.89,.98]	**.87**	**.78** [.67,.87]

Table 11: Results by case type. DAG advantage holds on both subsets.

Steps	ΔFDRec	ΔRCA	n traces
$\leq$3	+15 pp	+22 pp	127
4–5	+21 pp	+33 pp	198
$\geq$6	+28 pp	+42 pp	125

Table 12: DAG advantage by workflow length.

H Per-Workflow Breakdown

Wkfl	Method	FDRec	FPR	HA	RCA
CS	E2E	.35	.07	.48	N/A
	Flat	.63	.17	.68	.33
	Rule	.54	.04	.60	.41
	AE	**.92**	.08	**.86**	**.75**
DA	E2E	.43	.09	.54	N/A
	Flat	.69	.14	.73	.42
	Rule	.60	.05	.65	.49
	AE	**.88**	.07	**.85**	**.78**
DP	E2E	.46	.08	.55	N/A
	Flat	.70	.13	.72	.40
	Rule	.61	.06	.64	.46
	AE	**.86**	.06	**.81**	**.64**

Table 13: Per-workflow breakdown.

Performance on non-DAG traces ($\sim$12%, 54 traces total): FDRec 0.82 [0.73, 0.90] and RCA 0.58 [0.46, 0.69], lower than overall results but above Flat Step (FDRec 0.64, RCA 0.35 on the same subset). Degradation concentrates in retry-loop traces;

dynamic branching traces show smaller degradation (FDRec 0.85).

I Judge Model Sensitivity

Judge Model	FDRec	FPR	HA (κ)	RCA
GPT-4o	**.89**	.07	**.84**	**.72**
Claude 3.5 Sonnet	.87	.09	.81	.69
GPT-4o-mini	.83	.11	.77	.65
Llama 3 70B	.80	.13	.73	.61

Table 14: Judge sensitivity. DAG advantage over Flat Step remains significant ($p < 0.01$) for all judges.

J Agent–Judge Independence

Workflow	Judge	FDRec	HA	RCA
CS (Claude agent)	GPT-4o	.92	.86	.75
	Claude 3.5 Sonnet	.90	.83	.73
DA (Claude agent)	GPT-4o	.88	.85	.78
	Claude 3.5 Sonnet	.86	.82	.75
DP (Llama agent)	GPT-4o	.86	.81	.64
	Claude 3.5 Sonnet	.85	.80	.63

Table 15: Same-family judging produces comparable or lower scores, ruling out self-evaluation inflation.

K Architecture Scope Details

To characterize the practical boundary of DAG-based evaluation, we simulate increasing non-DAG rates by randomly converting DAG traces to non-DAG (loop/branch) structures. At 30% non-DAG rate, FDRec remains 0.83 and RCA 0.63; at 50%, FDRec is 0.78 and RCA 0.55, still above flat evaluation but with diminished advantage. The practical boundary is approximately 60% non-DAG rate, beyond which the DAG advantage becomes marginal (<5 pp). AGENTEVAL is most valuable for sequential and moderately-branching architectures, which represent the majority of current production deployments (Table 16).

Architecture Class	Est. Non-DAG %	Source
Sequential tool-calling	8–15%	Internal agents
Moderate branching	20–30%	τ-bench
SWE agents (edit-test loops)	25–35%	SWE-bench
Multi-agent collaboration	40–60%	Estimated†
Tree-of-thought / LATS	60–80%	Estimated†

Table 16: Estimated non-DAG trace rates. †Based on architectural analysis, not empirically validated.

L Regression Case Studies

Three illustrative cases from the pilot: (1) A prompt template update for CS-Agent replaced a structured instruction with a free-form variant. AGENTEVAL flagged a drop in PARAMGEN quality for account-lookup ($p < 0.01$, FDRec dropped 12 pp), which propagated downstream. The team reverted within hours. (2) DA-Agent's SQL generation degraded after an LLM provider-side update; AGENTEVAL localized the regression to code generation steps, enabling a targeted fix. (3) A DP-Agent schema change improved average-case performance by 2 pp but degraded long-document handling by 11 pp, detected only by the edge-case tier of progressive evaluation.

M Deployment Methodology Details

Baseline: Self-reported surveys from 12 engineers covering 47 incidents before adoption. Median: 4.2 hours; P25: 1.8 hours; P75: 8.5 hours; P90: 14 hours. Self-reports may overestimate (recall bias) and capture total time including communication overhead.

With AGENTEVAL: System-logged times from 156 events. Median: 22 minutes; P25: 8 minutes; P75: 48 minutes; P90: 1.6 hours. Logged times capture alert-to-first-code-change interval and may *underestimate* by excluding initial investigation.

Both measurements consistently show substantial improvement across all three teams, but the exact magnitude should be interpreted with caution given measurement asymmetry and confounds (failure severity, Hawthorne effects, concurrent improvements).

Additional deployment impact: For DA-Agent, taxonomy-guided analysis identified "parameter hallucination" (syntactically valid but semantically incorrect SQL WHERE clauses) as 31% of execution failures. Targeted few-shot examples reduced DA-Agent's parameter error rate from 27% to 11%.

Onboarding a new workflow requires ∼20–30 person-hours. Same-domain workflows require less (∼12–18 hrs) through partial reuse. Preliminary experiments suggest GPT-4o-assisted DAG schema proposals reduce setup time by approximately 40%.

N Deployment Lessons Learned

Five insights from deployment:

First, taxonomy-first design is essential. After the first month, 34% of CS-Agent failures were

"parameter hallucination" errors not captured by initial metrics. *Lesson*: Start with failure modes, not metrics.

Second, calibration is critical and brittle. Uncalibrated judges showed 15% lower agreement. Five stratified anchors per metric was the minimum; random anchors underperformed due to skewed score distributions.

Third, schema-defined and trace-inferred DAGs serve complementary purposes. Structural deviation proved a useful quality signal: deviant traces were associated with 2.1× higher failure rates.

Fourth, progressive evaluation changes behavior. Teams that adopted smoke tests ran 3.4× more evaluation cycles per sprint.

Fifth, taxonomy maintenance is a sociotechnical challenge. We added 3 subcategories during the pilot and adopted a "propose-discuss-merge" protocol for governance.

O Extended Limitations

Methodological Boundaries. Our evaluation dataset is sourced from anonymized logs using production tools and prompts but does not reflect full live-system complexity. The reported $\kappa = 0.84$ reflects agreement against majority-voted references; per-annotator agreement ($\kappa = 0.78$–0.86, Table 18) provides a more conservative estimate. The 150 human-annotated cases provide adequate power for aggregate metrics but wide per-workflow CIs. Ground truth for internal evaluation was produced by experts separate from authors; τ-bench and SWE-bench annotations were produced by authors with independent validation on a subset. Cost analysis assumes January 2026 API pricing.

Generalization Scope. Our taxonomy was derived from three workflow types and may not cover other domains; cross-system evaluation suggests 76% taxonomy coverage on out-of-domain agents. We evaluate English-language workflows only. The 5-type step classification covers common tool-calling patterns but may need extension for specialized architectures.

P Threshold Sensitivity Analysis

Threshold shift	FDRec	FPR	RCA
$\theta - 0.5$ (more sensitive)	.92	.14	.69
θ (default)	.89	.07	.72
$\theta + 0.5$ (more conservative)	.85	.04	.67

Table 17: Sensitivity to uniform threshold perturbation (± 0.5).

Per-type thresholds: $\theta_{\text{PLAN}} = 3.0$, $\theta_{\text{TOOLSEL}} = 3.0$, $\theta_{\text{PARAMGEN}} = 2.5$, $\theta_{\text{EXEC}} = 3.0$, $\theta_{\text{SYNTH}} = 3.0$, selected by grid search on a held-out 52-trace subset. PARAMGEN thresholds are most sensitive: ± 0.5 shifts FDRec by ± 4 pp, compared to ± 2 pp for other types.

Aggregation Function. The workflow-level score is the weighted harmonic mean:

$$Q(\mathcal{G}) = \frac{\sum_i w_i}{\sum_i w_i / q(v_i)}, \quad w_i = |\text{desc}(v_i)| + 1 \quad (1)$$

where $\text{desc}(v_i)$ is the set of descendants of v_i.

Q Counterfactual Analysis

To demonstrate counterfactual analysis, we replace a CS-Agent root cause (v_2, tool selection) with the gold-reference output and re-evaluate downstream. Scores improve: v_3 from 2.1 to 4.6, v_4 from 1.8 to 4.3, v_5 from 2.3 to 4.1. Across 30 sampled failing traces, counterfactual analysis confirms AGENTEVAL-identified root causes in 87% of cases. This measures the combined effect of error correction and judge sensitivity to improved context.

R Systematic Disagreement Patterns

Analyzing 156 step evaluations where AGENTEVAL disagrees with human majority vote: (1) *partial correctness* (43%): steps with partially correct output that humans rate leniently but the judge rates strictly, especially SYNTH steps; (2) *context sensitivity* (31%): cases where human evaluators use external knowledge not in the trace; (3) *borderline thresholds* (26%): steps near the pass/fail boundary where judgments diverge by exactly 1 point.

S Parsing Failure Analysis

Of 21 total parsing failures (12 τ-bench + 9 SWE-bench): multi-agent handoffs without explicit dependency (8 traces), unbounded loops exceeding the 5-iteration unrolling limit (7 traces), and deeply nested conditional branches (6 traces).

T Error Propagation Statistics

Across all 450 test cases, 847 total step-level failures: 312 (36.8%) root cause failures, 535 (63.2%) propagated failures. Average propagation chain length: 2.1 steps. Context loss failures have the longest average chain (3.2 steps); API/tool failures the shortest (1.4 steps).

U Per-Annotator Agreement

Annotator Pair	κ (vs. AgentEval)	κ (pairwise)
Annotator 1 (CS)	0.86	0.81
Annotator 2 (CS)	0.83	0.78
Annotator 3 (CS)	0.82	0.77
Annotator 4 (DA)	0.85	0.80
Annotator 5 (DA)	0.84	0.79
Annotator 6 (DA)	0.81	0.76
Annotator 7 (DP)	0.80	0.78
Annotator 8 (DP)	0.78	0.75
Annotator 9 (DP)	0.79	0.77

Table 18: Per-annotator agreement.

V Per-Metric Human Agreement

Step Type	Metric	κ
PLAN	Completeness	0.78
PLAN	Feasibility	0.81
TOOLSEL	Selection accuracy	0.92
TOOLSEL	Tool relevance	0.85
PARAMGEN	Param. correctness	0.89
PARAMGEN	Param. completeness	0.87
SYNTH	Faithfulness	0.80
SYNTH	Completeness	0.76
SYNTH	Coherence	0.82
Average		*0.83*

Table 19: Per-metric human agreement (single-judge GPT-4o).

W Score Distribution and Confusion Matrix

Score	1	2	3	4	5
Human (%)	8.2	11.5	14.8	28.7	36.8
AgentEval (%)	7.6	11.5	16.3	29.4	35.2

Table 20: Score distribution across 150 human-annotated cases (step-level).

Human	AgentEval 1	2	3	4	5	Total
1	71	8	2	0	0	81
2	4	93	15	2	0	114
3	0	11	112	23	0	146
4	0	2	28	228	25	283
5	0	0	4	37	322	363
Total	75	114	161	290	347	987

Table 21: Confusion matrix (987 step evaluations). Binary pass/fail agreement (threshold 3.0) is 94.0%.

X DAG Example Figure

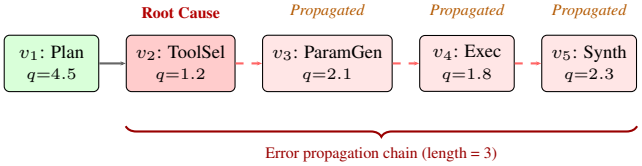

Figure 2: DAG-based evaluation of a CS-Agent workflow. A wrong tool selection at v_2 (root cause, $q = 1.2$) propagates through v_3–v_5.

Y Failure Mode Distribution

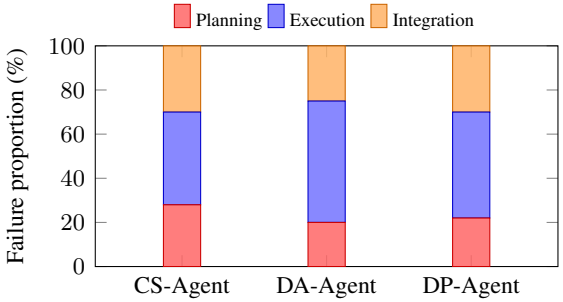

Figure 3: Failure mode distribution across workflows.