# TRIAGE: Severity-Ranked Multi-Agent Failure Attribution

**Anonymous ACL submission**

## Abstract

Failure attribution for multi-agent LLM systems is typically formulated as identifying a single decisive error in a failed execution trace. This formulation is brittle: failed traces often contain multiple interacting errors, and the decisive step is contested both among human annotators and between humans and frontier-LLM judges, a disagreement we quantify in TriageBench. We therefore reframe failure attribution as *severity-ranked diagnostic triage*, returning a ranked list of intervention candidates rather than a single blamed step, and instantiate it as TRIAGE, a zero-shot cost-aware cascade combining LLM-as-judge suspicion scoring, a lightweight structural-impact graph, and LLM-simulated replay over generated repair patches. The suspicion stage alone ($\rho$, a single LLM call that scores every step instead of picking one) already reaches 41.8% top-1 on Who&When, +14.0pp over the strongest prior prompting baseline. Under top-$K$ evaluation, the full TRIAGE pipeline reaches 73.9% top-5 on Who&When (vs. 42.4% top-1) and 93.2% top-10 on multi-error TRAIL (vs. 56.8% top-1); its top-3 contains at least one plausible repair on 98.5% / 99% of 326 traces (human rater / 3-LLM panel) under lenient (good + ok) rating.

## 1 Introduction

Multi-agent LLM systems coordinate specialized agents that plan, call tools, and verify each other's outputs across long-horizon executions, tackling tasks no single LLM call can complete (Wu et al., 2024; Hong et al., 2024; Fourney et al., 2024). When such a system fails, the wrong final answer is rarely the whole story. Multiple agent actions typically contribute, with errors compounding across the planning, execution, and verification roles distributed over different agents (Cemri et al., 2026). Diagnosing which of these actions caused the failure is therefore a prerequisite for debugging and reliable deployment of these systems (Epperson et al., 2025; Solomon et al., 2026). The Who&When benchmark formalized this as single-label prediction over a failed *trace* (the recorded sequence of agent messages, tool calls, and tool outputs from one execution), across 184 such traces. The prediction target is the *decisive error step*, the action whose correction would, counterfactually, have led the rest of the trajectory to the correct final answer (Zhang et al., 2025b). A range of methods has built on this definition, including iterative judging (Zhu et al., 2026), hierarchical causal graphs (Wang et al., 2026), and consensus voting over hierarchical context (Banerjee et al., 2025). Yet step-level accuracy plateaus around 40% for the strongest methods, raising the question of whether the bottleneck lies in the methods or in the single-label formulation itself.

Evidence points to the latter. Failed traces often contain multiple plausibly contributing steps. Who&When's own human annotators disagree on the decisive error step in 9.1 to 50% of traces, and our frontier-LLM council disagrees with the original annotators on the decisive step in 59.3% of traces (Section 7). As an illustration, consider a Who&When trace from our TriageBench study. A user requests filling ingredients from a strawberry-pie audio recipe. A transcription agent fabricates "salt" and misses "vanilla extract"; a culinary agent alphabetizes the wrong list; a verifier approves it. Who&When's annotator blames the culinary agent for keeping the fabricated "salt"; our council blames the transcription that produced it. Both are fixable: re-transcription or a cross-check at the culinary or verifier step would each correct the trace. Attribution should therefore not commit to one accusation but return a ranked list of intervention candidates, ordered so practitioners inspect the most consequential steps first. No existing method produces such a ranking.

The reformulation alone is empirically powerful. TRIAGE's suspicion stage ($\rho$) is a single LLM

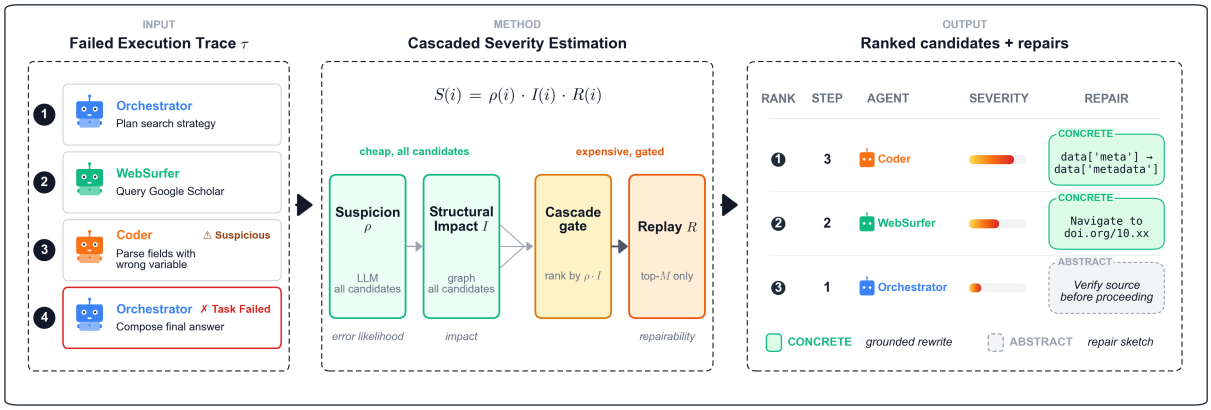


Figure 1: TRIAGE pipeline. Each step gets suspicion ($\rho$), structural impact ($I$), and replay-based uplift ($R$); $\rho$ and $I$ are computed for all candidates while patches and replay are gated to the top-$M$. Output: a severity-ranked list of intervention candidates with paired repair patches.

call that scores every step in $[0, 1]$ and ranks by score, replacing single-label prompts that commit to one decisive step. On Who&When (Kimi K2.5), it lifts top-1 accuracy to 41.8%, a +14.0pp gain over the strongest published prompting baseline (Step-by-Step, 27.8%) and within 0.6pp of the full cascade (Section 6.1). Top-1 against a single gold label appears at least partly annotation-limited, so the discriminative work moves to top-$K$ coverage: Who&When accuracy rises from 42.4% top-1 to 73.9% top-5, and on multi-error TRAIL reaches 93.2% top-10 vs. 56.8% top-1 (Section 6.2).

Building on this finding, we propose TRIAGE, a zero-shot cost-aware diagnostic cascade. We define a step's severity as its *intervention value* (Section 3), scored by three per-step signals: the suspicion score $\rho$ above, a structural-impact graph score $I$ estimating downstream propagation at zero LLM cost, and a replay uplift $R$ from an LLM-simulated replay of a generated repair patch. Patch generation and replay are gated to the top-$M$ candidates ranked by $\rho \cdot I$. TRIAGE's top-3 set contains at least one plausible repair on 99% of 326 rated traces under a lenient threshold (one human rater and a 3-LLM panel, rating each candidate independently with access to the gold answer; Section 6.4), turning attribution from localization into an actionable repair plan.

Our contributions are: (1) we reformulate multi-agent failure attribution as severity-ranked diagnostic triage, motivated by annotator disagreement on the decisive step, and show that top-$K$ evaluation reveals substantially more recoverable signal than top-1; (2) we instantiate this as TRIAGE, a zero-shot cascade that pairs each top-$M$ candidate with a generated repair patch and a simulated-replay uplift estimate; and (3) we release TriageBench, a frontier-LLM council's continuous severity scores over 1,184 candidate intervention targets on Who&When, evidence that the top-1 ceiling is at least partly annotation-limited.

## 2 Related Work

### 2.1 MAS Failure Modes and Taxonomies

Cemri et al. (2026) analyzed 1,600+ annotated traces across seven MAS frameworks and built MAST-Data, a taxonomy of 14 failure modes spanning specification/system design issues, inter-agent misalignment, and task verification. Other work shows that tracing and correcting root causes recovers measurable end-task performance (Zhu et al., 2025).

### 2.2 Post-hoc Failure Attribution

The Who&When benchmark formulated automated failure attribution, defining the task as identifying the earliest decisive error whose correction would flip a failed trace to success (Zhang et al., 2025b). The original Who&When baselines (all-at-once, step-by-step, binary search) are prompting strategies that reach only 12.3% step accuracy in the source paper; reproduced on the stronger Kimi K2.5 model, step-by-step rises to 27.8% (Table 1), still well below TRIAGE's 41.8% ($\rho$-only) and 42.4% (full cascade) on the same model (Section 6.1).

Subsequent methods span several directions. One line builds structured representations of the trace before attribution: ECHO decomposes traces into hierarchical context levels (Banerjee et al.,

2025), and CHIEF constructs causal graphs over agent interactions to propagate blame (Wang et al., 2026). RAFFLES iterates instead, employing aspect-specific evaluators that narrow candidates over successive judging rounds (Zhu et al., 2026). A third line reduces the search space before localization: SDBL delineates a failure-containing scope before reasoning within it (Sun et al., 2026), while CDC-MAS combines Shapley values with performance causal inversion on an 80/10/10 split of Who&When (Ma et al., 2025). Other approaches bring in signals beyond the trace (Ge et al., 2025; Kong et al., 2026).

TRAIL (Deshpande et al., 2025) reframes the task at the benchmark level: rather than annotating one decisive error per trace, it labels every error with a fine-grained taxonomy, reporting an average of 5.7 errors per failed trace across 148 executions. TRIAGE explores a complementary reframing, ranking candidate steps by repair severity rather than enumerating all errors with equal weight.

### 2.3 Counterfactual and Replay-Based Approaches

Most methods invoke the counterfactual definition of a decisive error (correcting the step averts failure) but apply it with varying fidelity. DoVer performs faithful re-execution from checkpoints, showing gains on hand-crafted datasets but requiring full system access that stored-trace benchmarks lack (Ma et al., 2026a). Other approaches simulate counterfactual reasoning in a single LLM call (West et al., 2025) or use offline replay to train lightweight LLM models (Zhang et al., 2025a). TRIAGE generates a repair patch for each candidate and scores it through blinded LLM comparison on stored traces, producing a per-candidate uplift estimate. It requires neither system access nor training data while producing inspectable repair hypotheses.

## 3 Task Formulation

**Background.** We adopt the turn-based multi-agent protocol of Zhang et al. (2025b). A system with $N$ agents produces a length-$T$ trajectory $\tau = (a_1, a_2, \ldots, a_T)$, where each action $a_t$ is taken by agent $\phi(t) \in \{1, \ldots, N\}$ from a per-agent action space $A_{\phi(t)}$. Let $Z(\tau) \in \{0, 1\}$ be the failure indicator, with $Z(\tau) = 1$ if the trajectory fails to solve the task.

For a failed trajectory, step $i$ is a *decisive error* if there exists an alternative action $\tilde{a}_i \in A_{\phi(i)}$ such that the modified trajectory $\tau'$, obtained by replacing $a_i$ with $\tilde{a}_i$, satisfies $Z(\tau') = 0$. Multiple decisive errors may co-exist in a single trajectory. Who&When's task is to identify the earliest such step,

$$t^* = \min\{i : i \text{ is a decisive error}\}. \quad (1)$$

Rather than predicting a single $t^*$, we generalize the task by ranking all steps of $\tau$ by *severity*, defined as the intervention value of a step.

**Definition 1** (Severity)**.** *For step $i$ in a failed trajectory $\tau$, severity is the maximum probability that an alternative action at $i$ would avert the failure:*

$$S(i) := \max_{\tilde{a}_i \in A_{\phi(i)}} \Pr[Z(\tau') = 0], \quad (2)$$

*where $\tau'$ is obtained by replacing $a_i$ with $\tilde{a}_i$, and the probability is taken over the stochastic rollout of subsequent steps.*

In the deterministic limit, $S(i) > 0$ if and only if step $i$ is a decisive error; severity is the continuous generalization of Who&When's binary indicator. Definition 1 casts severity as an interventional quantity in the sense of Pearl (2009): the maximum over alternative actions is the counterfactual best-case probability of averting failure under a single-step intervention. Ranking candidates by such suspiciousness scores has a long tradition in software fault localization, where Tarantula (Jones et al., 2002) and the Ochiai coefficient (Abreu et al., 2007) rank program elements by their statistical association with observed failures; we lift this paradigm to LLM-agent traces.

The task output is a ranked list

$$\mathcal{R} = \{(a_i, S(i), r_i)\}_{i=1}^{T}, \quad (3)$$

sorted by decreasing $S(i)$, where $r_i$ is a candidate repair for step $i$. The top-ranked candidate inherits the single-step interpretation of $t^*$; the full list provides a prioritized repair plan. Since $S(i)$ depends on quantities we cannot observe (the optimal alternative and the rollout distribution), TRIAGE estimates it via the cascaded estimator introduced in Section 4.

## 4 Method

TRIAGE approximates the severity ranking induced by $S(i)$ with a continuous score that decomposes into three diagnostic factors:

$$\hat{S}(i) = \rho(i) \cdot I(i) \cdot R(i) \quad (4)$$

where $\rho(i) \in [0,1]$ estimates the probability that step $i$ is erroneous, $I(i) \in [0,1]$ estimates the structural propagation of that error, and $R(i) \geq 1$ is a per-candidate uplift estimate from replay. We treat $\hat{S}$ as a pragmatic repair-priority ranker, not a calibrated estimator of the causal severity $S(i)$ from Definition 1. Under signal independence, $\rho \cdot I$ approximates the decisive-error probability; $R$ is promotion-only ($R \geq 1$) and applied only to top-$M$ candidates, so it serves primarily as validation rather than as a strong re-ranker (Section 6.3). Section 6.3 decomposes each factor's empirical contribution.

The pipeline computes $\hat{S}$ in four stages: trace normalization, suspicion scoring ($\rho$), structural impact ($I$), and repair generation with blinded replay ($R$) over a top-$M$ subset. Candidates are ranked by $\hat{S}$; the top-ranked candidate is the decisive-error prediction, and the full ranking is the prioritized repair plan.

**Trace normalization.** TRIAGE normalizes the raw trace into a uniform step sequence $\tau = (a_1, \ldots, a_T)$, treats every normalized step as a candidate ($\mathcal{C} = \{1, \ldots, T\}$), and annotates each step with its producing agent $\phi(i)$ and a structural role inferred by a rule-based classifier (LLM response, tool call, planning step, etc.; Appendix A.1).

### 4.1 Suspicion Scoring ($\rho$)

$\rho$ instantiates our LLM-as-judge reformulation (Zheng et al., 2023): the LLM receives the normalized trace and assigns a suspicion score $\rho^{(p)}(i) \in [0,1]$ to every step in a single pass. We draw $P$=5 independent samples and average, following the self-consistency variance-reduction strategy of Wang et al. (2023):

$$\rho(i) = \text{clip}\left(\frac{1}{P}\sum_{p=1}^{P} \rho^{(p)}(i),\ 0.01,\ 1\right). \quad (5)$$

### 4.2 Structural Impact ($I$)

**Trace-impact graph.** For each candidate $i$, we estimate downstream impact at zero LLM cost from a heuristic trace-impact graph, adapting program-slicing intuitions (Weiser, 1984) to LLM-agent traces. A step's impact is determined by which downstream computations depend on its output. The graph identifies a bounded set of structural anchors downstream of $i$ (next agent response, next tool call, next tool result, next planning step, final conclusion), weights each by a logarithmic distance decay and a hand-designed step-type interaction prior, and amplifies anchors that are themselves suspicious. The aggregated graph score $I_{\text{graph}}(i) \in [0,1]$ captures how much an error at $i$ would propagate through subsequent execution. Full equations, the step-type weight table, and the anchor-selection rules are in Appendix B.

**Position-prior mix.** The graph score collapses on traces with shallow dependency structure (e.g., short retrieval traces with few downstream anchors). To stabilize ranking in this regime, we mix $I_{\text{graph}}$ with a position-decay prior $\pi$:

$$I(i) = \alpha I_{\text{graph}}(i) + (1-\alpha)\,\pi(i), \quad \pi(i) = \tfrac{1}{1+i}. \quad (6)$$

Intuitively, earlier interventions give the rest of the trajectory more chances to recover the correct outcome, while late interventions often inherit failures already locked in by upstream errors; the harmonic $\frac{1}{1+i}$ encodes this as a smooth monotone position prior, consistent with the compounding-error analysis of Ross et al. (2011) for sequential prediction.

### 4.3 Repair Generation and Replay ($R$)

This stage adapts the generate-and-validate paradigm of automated program repair (Le Goues et al., 2012) to LLM-agent traces, replacing test-suite execution with a blinded LLM evaluator.

**Replay budget.** Replay is the cascade's expensive stage, so we restrict it to the top-$M$=3 candidates by $\rho(i) \cdot I(i)$, denoted $\mathcal{C}_M = \text{Top-}M_{i \in \mathcal{C}}\, \rho(i) \cdot I(i)$, following the cost-aware cascade pattern in which cheap signals filter the candidate pool before expensive evaluators (Viola and Jones, 2001). Each candidate in $\mathcal{C}_M$ is scored under a single *replay perspective*, the rubric the replay judge applies when evaluating the patched trace (we use *critical error detection*; Table 6 lists ten interchangeable choices). Replaying under all 10 perspectives instead of one matches the single-perspective top-1 on 91% of traces (Appendix E.8), consistent with replay's role as per-candidate validation rather than re-ranking (Section 6.3). With typical candidate pools of 17 to 21 steps, full replay ($M$=$|\mathcal{C}|$, one perspective) would issue 34 to 42 calls per trace; the $M$=3 gate cuts this to 6, a >80% reduction.

**Taxonomy-conditioned patch generation.** For each $i \in \mathcal{C}_M$, repair generation proceeds in two LLM calls: a *hypothesis* call classifies the

candidate's error into one of 13 categories (Appendix C.1, derived from the MAST failure-mode taxonomy of Cemri et al., 2026), and a *patch* call applies a constrained operator paired with the inferred category to generate the repair at temperature 0. Generated patches are checked against per-operator Levenshtein-ratio caps and role-consistency rules (Appendix C). When a step resists grounded editing (e.g., a web-browsing step whose output depends on live search results), TRIAGE falls back to an abstract repair: a schematic description of the intended change (e.g., "use DOI link instead of search"), ensuring every candidate in $\mathcal{C}_M$ receives a testable intervention.

**Blinded LLM-simulated replay.** "Replay" here is LLM-simulated scoring of a patched trace, not re-execution of the multi-agent system. For each $i \in \mathcal{C}_M$, a blinded LLM evaluator estimates the success probability of the trajectory at step $i$ under two conditions: a baseline $p_0(i)$ from the original trace, and a patched estimate $\hat{p}(i)$ from the same trace with repair $r_i$ substituted. The evaluator sees only one version per call, with no indication of which. Execution-grounded validation (re-running agents under the patched step) is left for future work (see Limitations). The repair uplift is:

$$R(i) = \begin{cases} 1 + \max\left(0, \hat{p}(i) - p_0(i)\right) & i \in \mathcal{C}_M, \\ 1 & i \notin \mathcal{C}_M. \end{cases} \tag{7}$$

Replay is promotion-only: a failed replay is ambiguous because the patch may be incomplete or the evaluator insensitive even when the step is causal, so we cap $R \geq 1$. Appendix C.4 compares this against an unfloored variant.

**Hyperparameters.** Hyperparameters ($P$=5, $\alpha$=0.7, suspicion floor 0.01, anchor cap $|A(i)|\leq 6$) are set by inspection on Who&When and held fixed across all benchmarks and models. Patch generation runs at temperature 0, suspicion and replay at default; Levenshtein caps, graph weights, and formula ablation are in Appendices C, B, E.6.

# 5 Experiments

## 5.1 Datasets

**Who&When.** The Who&When benchmark (Zhang et al., 2025b) provides 184 failed multi-agent execution traces on GAIA (Mialon et al., 2024) and AssistantBench (Yoran et al., 2024) queries, partitioned into two splits by the multi-agent system that produced the trace. The Algorithm-Generated (AG) split contains 126 traces from CaptainAgent (Song et al., 2025), which builds GPT-4o agent teams per query. The Hand-Crafted (HC) split contains 58 traces from Magentic-One (Fourney et al., 2024), a fixed five-agent architecture. Each trace consists of the full conversation log, annotated with the agent responsible for the failure, the decisive error step, and a natural-language explanation. HC traces are approximately six times longer than AG traces on average (50 vs. 8.6 steps) and harder to attribute.

**TRAIL.** TRAIL (Deshpande et al., 2025) contains 148 traces (117 GAIA, 31 SWE-Bench-Lite (Jimenez et al., 2024); hereafter SWE-Bench) with 841 expert-annotated errors. Unlike Who&When, TRAIL annotates all errors per trace (mean 5.7). GAIA traces are shorter information-retrieval tasks; SWE-Bench traces are long code-debugging sessions whose mean input length (∼600K tokens) exceeds common frontier-model context limits. We compare on TRAIL's localization metric only; our 13-class error taxonomy (Appendix C.1) does not map cleanly to TRAIL's schema for joint evaluation.

## 5.2 Metrics

We report *step accuracy* (top-1, following Zhang et al., 2025b; Zhu et al., 2026), *top-$K$ recall* for ranked output, and *NDCG@3* for the component ablation; MRR, within-2, and agent-level metrics are in the appendix. Ties are broken by step index. Top-$K$ recall is not directly comparable across the two datasets: Who&When credits a hit only when the single gold step appears in the top-$K$, whereas TRAIL credits any of the multiple annotated errors per trace as a hit.

## 5.3 Baselines

**Prior methods.** On Who&When, we compare against published step-accuracy results from RAFFLES (Zhu et al., 2026), CHIEF (Wang et al., 2026), SDBL (Sun et al., 2026), ECHO (Banerjee et al., 2025), and CDC-MAS (Ma et al., 2025), taking each method's best per-subset configuration. On TRAIL, we take the strongest LLM-as-judge result per subset across the eight reported LLMs; Gemini-2.5-Pro is the strongest on both GAIA (54.6% top-1) and SWE-Bench (23.8% top-1).

**Same-LLM baselines.** To isolate method from model, we re-run Who&When's three published

| Method | Llama 3.3 70B | | | Kimi K2.5 | | | Claude Sonnet 4 | | |
|---|---|---|---|---|---|---|---|---|---|
| | AG | HC | All | AG | HC | All | AG | HC | All |
| All-at-Once | 15.9 | 5.2 | 12.5 | 23.0 | 8.6 | 18.5 | 16.7 | 3.4 | 12.5 |
| Step-by-Step | 7.1 | 8.6 | 7.6 | 31.8 | 19.0 | 27.8 | 9.5 | 0.0 | 6.5 |
| Binary Search | 14.3 | 10.3 | 13.0 | 24.6 | 20.7 | 23.4 | 19.8 | **13.8** | 17.9 |
| $\rho$-only | 31.7 | **15.5** | 26.6 | 51.6 | 20.7 | 41.8 | **51.6** | 10.3 | **38.6** |
| TRIAGE | **36.5** | 13.8 | **29.3** | **52.4** | **20.7** | **42.4** | 49.2 | 12.1 | 37.5 |

Table 1: Step accuracy (%) on Who&When across three LLMs. AG ($n$=126), HC ($n$=58), All ($n$=184 weighted; all traces included, method failures counted as misses). Bold/underline: best/second-best per column.

prompting strategies (Step-by-Step, Binary Search, All-at-Once) (Zhang et al., 2025b) on each LLM. We also report two single-call baselines: TRIAGE's suspicion stage in isolation ($\rho$-only, the per-step reformulation that scores every step in one pass; Appendix E.4), and *LLM-judge (target $N$=10)*, a single call that asks the LLM to enumerate the 10 most likely decisive errors as a ranked list (Appendix A.5). LJ $N$=10 adapts TRAIL's exhaustive-enumeration protocol by adding an explicit ranking instruction; both baselines are distinct from the full TRIAGE cascade that adds structural impact and replay.

### 5.4 Implementation Details

TRIAGE uses a single LLM for all stages and operates without access to the ground-truth decisive step or final answer at any stage of the pipeline (matching the setting of the same-LLM baselines and prior methods in Table 2). Primary: Kimi K2.5 (`kimi-k2.5` via Moonshot). Transferability runs: Llama 3.3 70B (`meta-llama/Llama-3.3-70B-Instruct-Turbo` via Together) and Claude Sonnet 4 (`claude-sonnet-4-20250514` via Anthropic, matching RAFFLES's model). Per-trace cost: Appendix F.

## 6 Results

This section reports top-1 accuracy across models and benchmarks (Section 6.1), top-$K$ coverage gains from ranked output (Section 6.2), what the cascade adds beyond ranking (Section 6.3), and a patch plausibility study (Section 6.4).

### 6.1 Top-1 Across Models

**Top-1 attribution on Who&When appears annotation-limited.** Table 1 reports our same-LLM reproductions, and Table 2 adds cross-paper literature context. Multi-stage methods cluster within ∼2pp on the All split despite spanning very different architectural complexity, namely TRIAGE (42.4% on Kimi K2.5), RAFFLES (42.4% on Sonnet 4), and CHIEF (40.5% on DeepSeek-V3.2). Removing the cascade leaves TRIAGE's per-step suspicion stage alone ($\rho$-only) at 41.8% on Kimi K2.5, inside the same band. This cluster comparison uses each method's best-reported configuration; on the shared Sonnet 4 model, TRIAGE reaches 37.5%, 4.9pp below RAFFLES. That low-40s plateau is consistent with an annotation-limited task: Who&When's own annotators disagree on 9.1 to 50% of traces, and the TriageBench council and the original annotators disagree on 59.3% (Section 7).

| Method | Model | AG | HC | All |
|---|---|---|---|---|
| ECHO | Cl. 3.5 Sonnet | 27.2 | 26.8 | 27.1 |
| CDC-MAS | GPT-4o | 36.2 | 18.2 | 30.5 |
| SDBL | DeepSeek-V3 | 39.7 | 27.8 | 35.9 |
| CHIEF | DeepSeek-V3.2 | 45.6 | **29.3** | 40.5 |
| RAFFLES | Claude Sonnet 4 | **51.6** | 22.4 | **42.4** |

Table 2: Prior reported step accuracy (%) on Who&When, each method's best-reported configuration. Bold/underline: best/second-best per column.

**The reformulation does the lifting; the cascade shifts top-1 little.** TRIAGE's suspicion stage ($\rho$) instantiates the reformulation via its per-step scoring prompt (Section 4). Across all three models, $\rho$-only and the full cascade are within 2.7pp at top-1 (Table 1; on Sonnet 4 $\rho$-only narrowly beats the cascade by 1.1pp), confirming that the cascade's role is to produce the diagnostic artifacts examined in Section 6.3, not to refine top-1 ranking. Cross-model, suspicion shows a recall-precision tradeoff (Appendix E.7): Sonnet 4 catches the gold step in top-$M$ more often but ranks within that pool less sharply than Kimi K2.5.

**Per-step ranking recovers headroom left by TRAIL's exhaustive-enumeration protocol.** TRAIL evaluates failure attribution by asking a single LLM call to enumerate every error in a trace

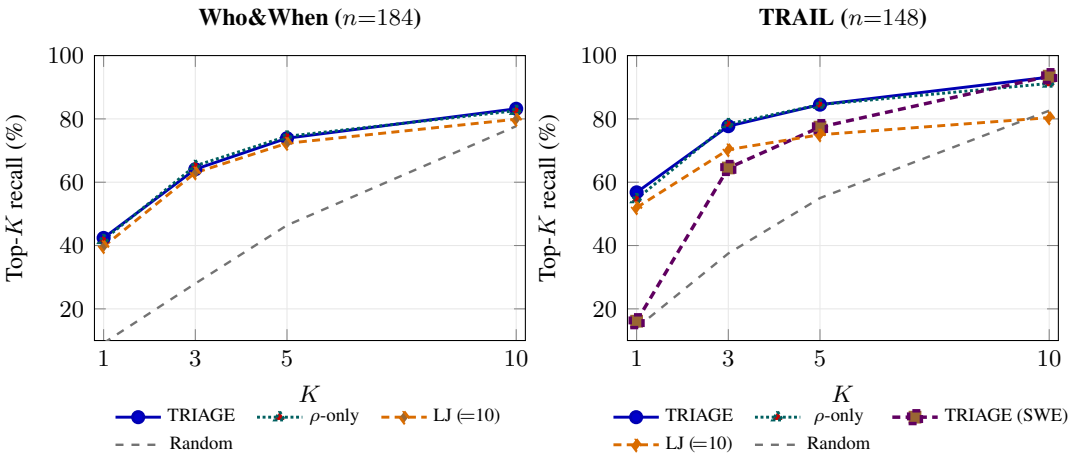


Figure 2: Top-$K$ step recall on Who&When ($n$=184) and TRAIL ($n$=148, any-error hit) using Kimi K2.5. $\rho$-only: TRIAGE's suspicion stage alone. LJ (=10): single-call top-10 ranked list. TRIAGE (SWE): SWE-Bench subset of TRAIL (31 traces).

as a flat list (Deshpande et al., 2025); traces contain ∼5.0 annotated errors on GAIA and ∼8.2 on SWE-Bench on average, and on long traces the output truncates so many errors go unreported. Reformulating as per-step ranking raises GAIA top-1 from 54.6% (Gemini-2.5-Pro, the strongest TRAIL-paper model) to 67.5% (TRIAGE on Kimi K2.5), a +12.9pp gain (Table 3); SWE-Bench is harder at top-1 but recovers at higher $K$, where TRIAGE's SWE-Bench top-5 reaches 77.4% against TRAIL's single-prediction 23.8%.

| Method | GAIA | SWE-Bench |
|---|---|---|
| TRAIL best (Gemini-2.5-Pro) | 54.6 | 23.8 |
| TRIAGE top-1 (Kimi K2.5) | 67.5 | 16.1 |
| TRIAGE top-5 (Kimi K2.5) | 86.3 | 77.4 |

Table 3: Per-subset top-$K$ step localization (%) vs. TRAIL paper-reported best-per-subset numbers (Deshpande et al., 2025).

### 6.2 Top-$K$ Recovery

**Top-$K$ reveals far more solvability than top-1 suggests.** Figure 2 shows the top-$K$ curves. On Who&When, allowing a ranked list rather than a single-step commitment substantially raises accuracy between $K$=1 and $K$=5 (1.7×): TRIAGE reaches 73.9% top-5 (Kimi K2.5) vs. 42.4% top-1, and 83.2% top-10. On multi-error TRAIL the pattern is stronger: 93.2% top-10 vs. 56.8% top-1, a +36.4pp gap. SWE-Bench top-1 is uniquely low (16.1%) because its long traces (mean 85 steps vs. ∼22 on Who&When) penalize single-step commitment, and recovers to 77.4% top-5 once the budget allows multiple candidates. The single-step framing prior work optimizes against masks most of the practically recoverable signal.

**Per-step scoring beats top-$K$ selection; the cascade layers add patches, not coverage.** For the top-$K$ comparison we use LJ target = 10, the closest single-call analog to TRAIL's exhaustive-enumeration protocol. TRIAGE's $\rho$ stage instead asks for a 0 to 1 suspicion score on every step in one call, then ranks steps by score. On TRAIL traces (mean 5.7 annotated errors per trace, max 20), $\rho$ alone reaches 91.2% top-10 vs. LJ=10's 80.4% (+10.8pp). Full TRIAGE's +12.8pp gain over LJ=10 (93.2% vs. 80.4%, $p$<0.001 paired bootstrap at $n$=144; Appendix E.5) thus comes mostly from $\rho$; the cascade layers contribute at most +2.1pp on TRAIL (T1) and +0.6pp on Who&When (T1) and trade off slightly at intermediate $K$ (none significant on their own, paired bootstrap all $p$>0.20).

### 6.3 Cascade Contribution

**The cascade is not a second ranker on the suspicion engine; it is the engineering that turns a ranking into a usable diagnostic artifact.** If $\rho$ already carries the ranking (Section 6.2), the natural question is what the rest of TRIAGE's cascade is for. Three deliverables that $\rho$ alone does not provide: a generated repair patch (plausibility in Section 6.4), a per-candidate uplift $\hat{p}-p_0$ from replay (validation below), and a 78% replay-call reduction over a full-replay variant (gated to top-$M$=3 instead of $M$=$|\mathcal{C}|$; scoped to replay only, not total cost). TRIAGE's total per-trace cost (≈13 LLM

calls: 1 suspicion + 6 patch + 6 replay; Appendix F) is higher than lightweight ranking baselines. That spend buys the per-candidate patches and uplift estimates those baselines cannot produce.

| | W&W | | TRAIL | |
|---|---|---|---|---|
| | T1 | T10 | T1 | T10 |
| $\rho$-only | 41.8 | 82.6 | 54.7 | 91.2 |
| + impact ($I$) | **42.4** | **83.2** | **56.8** | **93.2** |
| + replay ($R$) | 42.4 | 83.2 | 56.8 | 93.2 |

Table 4: Cascade decomposition (Kimi K2.5) on Who&When ($n$=184) and TRAIL ($n$=148), top-1 (T1) and top-10 (T10) step accuracy (%).

**Structural impact ($I$) is the cascade's LLM-free gating signal.** On Who&When, adding $I$ to $\rho$ moves top-1 by +0.6pp (Table 4), within sampling noise. On multi-error TRAIL, $I$ adds +2.1pp at top-1 and +2.0pp at top-10 (Table 4), where richer downstream propagation gives the graph more to capture. Beyond ranking, $I$ also gates which candidates receive patches: a late high-$\rho$ step is often a downstream symptom whose repair cannot undo upstream failure, while an earlier moderate-$\rho$ step with many dependents is the pivotal intervention point. $I$ encodes this propagation structure at zero LLM cost.

**LLM-simulated replay ($R$) counterfactually validates TRIAGE's top-1 repair on 76% of Who&When and 41% of TRAIL traces** (fraction with $\hat{p} > p_0$ under blinded replay). The lower TRAIL rate reflects multi-error structure: a single repair addresses only one of ∼5.7 annotated errors per trace. The $R(i) \geq 1$ clamp keeps replay as validation rather than re-ranking; top-1 changes occur on only 1.1% / 4.1% of traces.

### 6.4 Patch Plausibility

**Under lenient rating, the top-3 patch set contains a plausible repair on nearly all traces; under strict rating, on a majority.** A human rater and a 3-LLM panel rated patch plausibility on all 326 traces on a 4-point scale, with access to the gold answer and rating each candidate independently (Appendix G). Top-3 any-plausible is 98.5% / 99% (human / 3-LLM panel majority) under lenient (*good* + *ok*) and 57% / 84% under strict (*good*-only); top-1 lenient is 86% / 86% and top-1 strict 34% / 52% (Table 26). Inter-rater agreement is moderate (binary $\kappa$ 0.35 to 0.71, Grok driving the lower end). The 23pp gap between strict top-1 (34% human) and strict top-3 (57% human) directly motivates the ranked-output design: no single accusation is reliably the best repair, but a top-3 set typically contains one. TRIAGE thus closes the loop from severity ranking to actionable diagnostic artifact: each trace yields up to three intervention candidates, each paired with a generated repair and a replay-based uplift estimate.

## 7 Evidence for Annotation Ambiguity in Top-1 Attribution

To probe whether the top-1 headroom on Who&When sits in the labels or in the methods, we release TriageBench: 177 Who&When traces with a candidate pool (mean 6.7 per trace, 1,184 total) judged pairwise by a council of three frontier LLMs (same models as Section 6.4) and aggregated via a Davidson tie-aware Bradley-Terry model (Davidson, 1970). The council judges severity in hindsight, given the gold final answer that attribution methods never see, and BT aggregation averages out single-judge variance, listwise position bias, and partial-ordering inconsistency; together these approximate an upper bound on LLM severity judgments on Who&When. Even this best-case signal disagrees with the original annotators on 59.3% of traces (council picks a step 2.14 positions later on average), consistent with the view that decisive-step labels are contested rather than that either annotation source is correct. Construction, posterior diagnostics, and the full results table are in Appendix D.

## 8 Conclusion

Failure attribution for multi-agent LLM systems has plateaued near 42% top-1 across published methods. Our results suggest this ceiling is at least partly annotation-limited: suspicion alone matches the strongest methods across three models, and a frontier-LLM council disagrees with the original Who&When annotators on the majority of traces. Reframing the task as severity-ranked diagnostic triage recovers far more solvability than top-1 reporting suggests. Pairing each ranked candidate with a generated repair patch and a replay-based uplift estimate turns localization into a usable diagnostic artifact: TRIAGE's top-3 set contains a plausible repair on 98.5% of traces under independent rating. We instantiate this as a training-free cost-aware cascade that gates patch generation and replay to the top-$M$=3 candidates.

## Limitations

**LLM proxies in repair and replay.** Two TRIAGE components rely on LLM judgment rather than real execution: replay scores each patched trace with a blinded LLM judge (Section 4) instead of re-running the multi-agent system, and patch plausibility is rated by one human rater plus a 3-LLM panel (Section 6.4) rather than by deployment users or downstream task success. An execution-grounded variant (re-running the agents under each patched step and measuring actual task-success change) would replace the replay proxy and stress-test the plausibility study, and is a priority for future work.

**Annotation-disagreement evidence.** TriageBench's 59.3% council vs. human disagreement (Section 7) is computed on 177 Who&When traces with a method-agnostic candidate pool (mean 6.7 per trace). Both LLM judges and the original annotators are imperfect, so the finding should be read as evidence that decisive-step labels are contested; we make no claim that the LLM council is correct. A larger, human-curated candidate pool would let future work attribute disagreement to specific annotator and model failure modes.

**Cross-model baselines.** Several prior-method comparisons (RAFFLES on Sonnet 4, CHIEF on DeepSeek-V3.2, TRAIL's reported models) are cross-model, so any TRIAGE-vs-prior accuracy gap conflates method and model effects (Section 6.1). The top-$K$ coverage reformulation, however, is promising independently of which method or model instantiates it, and remains the paper's priority direction for the field.

**Heuristic design.** The trace-impact graph uses hand-designed edge weights, criticality scores, and the position-mix coefficient $\alpha$=0.7, calibrated by inspection on Who&When and held fixed across all other benchmarks and models. The uniform-weight ablation (Appendix E.6) shows the hand-tuned weights beat uniform weights by only 1.1pp top-1, so the cascade is not strongly weight-tuned; a learned impact estimator could nevertheless improve severity ordering further, at the cost of the zero-shot property we deliberately preserve.

## 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)*, pages 89–98.

Adi Banerjee, Anirudh Nair, and Tarik Borogovac. 2025. Where did it all go wrong? a hierarchical look into multi-agent error attribution. *Preprint*, arXiv:2510.04886.

Amine Barrak. 2025. Traceability and accountability in role-specialized multi-agent LLM pipelines. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pages 315–322. IEEE.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E Gonzalez, and Ion Stoica. 2026. Why do multi-agent LLM systems fail? *Advances in Neural Information Processing Systems*, 38.

Roger R. Davidson. 1970. On extending the Bradley-Terry model to accommodate ties in paired comparison experiments. *Journal of the American Statistical Association*, 65(329):317–328.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. 2025. TRAIL: Trace reasoning and agentic issue localization. *Preprint*, arXiv:2505.08638.

Will Epperson, Gagan Bansal, Victor C. Dibia, Adam Fourney, Jack Gerrits, Erkang Zhu, and Saleema Amershi. 2025. Interactive debugging and steering of multi-agent AI systems. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–15. ACM.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. 2024. Magentic-One: A generalist multi-agent system for solving complex tasks. *Preprint*, arXiv:2411.04468.

Yu Ge, Linna Xie, Zhong Li, Yu Pei, and Tian Zhang. 2025. Who is introducing the failure? automatically attributing failures of multi-agent systems via spectrum analysis. *Preprint*, arXiv:2509.13782.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

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*.

James A. Jones, Mary Jean Harrold, and John Stasko. 2002. Visualization of test information to assist fault localization. In *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, pages 467–477, New York, NY, USA. Association for Computing Machinery.

Fanqi Kong, Ruijie Zhang, Huaxiao Yin, Guibin Zhang, Xiaofei Zhang, Ziang Chen, Zhaowei Zhang, Xiaoyuan Zhang, Song-Chun Zhu, and Xue Feng. 2026. AEGIS: Automated error generation and attribution for multi-agent systems. *Preprint*, arXiv:2509.14295.

Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer. 2012. GenProg: A generic method for automatic software repair. *IEEE Transactions on Software Engineering*, 38(1):54–72.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024. Llms-as-judges: A comprehensive survey on llm-based evaluation methods. *Preprint*, arXiv:2412.05579.

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*.

Guoqing Ma, Zhu Jia, Hanghui Guo, Weijie Shi, Jiawei Shen, Jingjiang Liu, and Yidan Liang. 2025. Automatic failure attribution and critical step prediction method for multi-agent systems based on causal inference. *Preprint*, arXiv:2509.08682.

Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2026a. DoVer: Intervention-driven auto debugging for LLM multi-agent systems. *Preprint*, arXiv:2512.06749.

Xuyan Ma, Xiaofei Xie, Yawen Wang, Junjie Wang, Boyu Wu, Mingyang Li, and Qing Wang. 2026b. Demystifying the lifecycle of failures in platform-orchestrated agentic workflows. *Preprint*, arXiv:2509.23735.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*.

Hadar Mulian, Sergey Zeltyn, Ido Levy, Liane Galanti, Avi Yaeli, and Segev Shlomov. 2026. AgentFixer: From failure detection to fix recommendations in LLM agentic systems. *Preprint*, arXiv:2603.29848.

Kartik Nagavajara, Garvit Sapra, Rishav Hada, and Nikhil Pareek. 2025. AgentCompass: Towards reliable evaluation of agentic workflows in production. *Preprint*, arXiv:2509.14647.

Judea Pearl. 2009. *Causality*. Cambridge university press.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Xu Shen, Qi Zhang, Song Wang, Zhen Tan, Xinyu Zhao, Laura Yao, Vaishnav Tadiparthi, Hossein Nourkhiz Mahjoub, Ehsan Moradi Pari, Kwonjoon Lee, and Tianlong Chen. 2026. Metacognitive self-correction for multi-agent system via prototype-guided next-execution reconstruction. *Preprint*, arXiv:2510.14319.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Ron Solomon, Yarin Yerushalmi Levi, Lior Vaknin, Eran Aizikovich, Amit Baras, Etai Ohana, Amit Giloni, Shamik Bose, Chiara Picardi, Yuval Elovici, and Asaf Shabtai. 2026. LumiMAS: A comprehensive framework for real-time monitoring and enhanced observability in multi-agent systems. *Preprint*, arXiv:2508.12412.

Linxin Song, Jiale Liu, Jieyu Zhang, Shaokun Zhang, Ao Luo, Shijian Wang, Qingyun Wu, and Chi Wang. 2025. Adaptive in-conversation team building for language model agents. *Preprint*, arXiv:2405.19425.

Kai Sun, Wenqiang Li, Bo Dong, Yuxin Lin, Jingyao Zhang, and Bin Shi. 2026. Scope delineation before localization: A two-stage framework for enhancing failure attribution in multi-agent systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(39):33108–33116.

P. Viola and M. Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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

Mark Weiser. 1984. Program slicing. *IEEE Transactions on Software Engineering*, 10(4):352–357.

Andrew A. West, Yixuan Weng, Min Zhu, Zhen Lin, Zhiyuan Ning, and Yue Zhang. 2025. Abduct, act, predict: Scaffolding causal inference for automated failure attribution in multi-agent systems. *Preprint*, arXiv:2509.10401.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. 2026. Survey on evaluation of llm-based agents. *Preprint*, arXiv:2503.16416.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. 2024. AssistantBench: Can web agents solve realistic and time-consuming tasks? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8938–8968, Miami, Florida, USA. Association for Computational Linguistics.

Yifan Yu, Moyan Li, Shaoyuan Xu, Jinmiao Fu, Xinhai Hou, Fan Lai, and Bryan Wang. 2025. CORRECT: COndensed eRror RECognition via knowledge transfer in multi-agent systems. *Preprint*, arXiv:2509.24088.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. 2025a. AgenTracer: Who is inducing failure in the LLM agentic systems? *Preprint*, arXiv:2509.03312.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, and Yiran Chen. 2025b. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Proceedings of the International Conference on Machine Learning*.

Xinkui Zhao, Sai Liu, Yifan Zhang, Qingyu Ma, Guanjie Cheng, Naibo Wang, and Chang Liu. 2026. ProMAS: Proactive error forecasting for multi-agent systems using markov transition dynamics. *Preprint*, arXiv:2603.20260.

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. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Chenyang Zhu, Spencer Hong, Jingyu Wu, Kushal Chawla, Yuhui Tang, Youbing Yin, Nathan Wolfe, Erin Babinsky, and Daben Liu. 2026. RAFFLES: Reasoning-based attribution of faults for LLM systems. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7659–7688, Rabat, Morocco. Association for Computational Linguistics.

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. *Preprint*, arXiv:2509.25370.

## Appendix Roadmap

The appendix follows the pipeline order before turning to evaluation, cost, validation, and reproducibility. Section A gives the prompt and implementation details; Section B defines the structural-impact graph; and Section C describes the error taxonomy, patch operators, and replay floor.

The remaining sections document benchmark construction and supporting evidence: Section D covers TriageBench; Section E reports additional experiments; Section F breaks down cost; Section G reports the human and LLM-panel patch rating study; and Section H lists reproducibility details.

## A Prompts and Implementation Details

All pipeline stages share an identical *task-and-trace prefix* (normalized question and serialized trace) at the start of each prompt.

The prompt templates also support an optional `{groundtruth_answer}` field. This field is left unset in all main-paper pipeline runs (TRIAGE attribution and the LLM-as-judge baseline), so no pipeline stage receives the gold final answer. The TriageBench council (Section D.3) does receive the gold answer because its role is to judge severity given the known correct outcome, not to perform attribution. Prompts below are reproduced from the supplementary codebase, with the trace prefix elided as `{task_and_trace_prefix}`.

### A.1 Trace normalization

Raw traces (Who&When, TRAIL) are normalized into a uniform sequence of step records. Each record contains an integer `step_id`, agent name, structural role (`plan`, `tool_call`, `tool_result`, `factual_assertion`, `tool_interpretation`, `conclusion`, `delegation`, `other`), full content (no truncation by default), and optional metadata.

Every normalized step enters the candidate pool. Routing tokens, system delimiters, and user task descriptions are tagged but not removed, ensuring that any step labeled by an annotator as the decisive error remains reachable by the ranker. Step roles are inferred by a small rule-based classifier using regular expressions and role-keyword matching on the agent's message; the same classifier feeds the structural-impact graph. Table 5 reports post-normalization trace statistics.

| Split | $n$ | mean | median | p25 | p95 |
|---|---|---|---|---|---|
| W&W AG | 123 | 8.6 | 7 | 6 | 14 |
| W&W HC | 55 | 50.0 | 45 | 32 | 92 |
| W&W All | 178 | 22.2 | 10 | 7 | 72 |
| TRAIL GAIA | 114 | 18.3 | 16 | 10 | 43 |
| TRAIL SWE | 30 | 84.7 | 73 | 41 | 187 |
| TRAIL All | 144 | 31.7 | 19 | 12 | 92 |

Table 5: Normalized trace length (steps) per split. Denominators are traces with successful normalization (4 TRAIL traces dropped at the canonicalize stage); the headline accuracy tables in the body use the full benchmark denominators ($n$=184 W&W, $n$=148 TRAIL) with normalization/parse failures counted as misses.

### A.2 Suspicion scoring prompt

The suspicion prompt presents the normalized trace and asks the LLM to rate every step's likelihood of being the *root-cause* mistake on a [0,1] scale. Output is constrained to a single `SCORES:` line, parsed by regex with a JSON-object fallback. Scores are clipped to $[0.01, 1]$ to preserve a non-zero replay prior on every candidate. We run $P = 5$ independent passes and average per-step.

**Suspicion prompt**

```
{task_and_trace_prefix}
## Your role
You are an expert debugger analysing the multi-agent
      conversation above, which FAILED to solve the task.
      Your job is to rate how likely each step is to be
      the ROOT-CAUSE mistake, the earliest step whose
      error caused downstream failure.

## What to look for
- A step where an agent makes a factual error, uses a
      wrong tool argument, misinterprets a result, or
      chooses a bad strategy.
- Focus on the EARLIEST mistake (root cause), not on
      downstream symptoms.
- "tool_result" steps are outputs from tools; the mistake
      is usually in the "tool_call" or "reasoning" step
      BEFORE a bad tool result.
- "termination" steps just end the conversation; they are
      almost never the root cause.

## Instructions
1. For each step, assign a suspicion score between 0.0
      and 1.0.
2. Most steps should be low (0.0-0.2); only a few should
      be high.
3. Output JSON mapping step_id to score.
4. Use this exact format on the last line:
SCORES: {"0": 0.05, "3": 0.85, "7": 0.40}
```

The prompt includes three priors a human debugger applies by default but the LLM does not (`tool_result` steps reflect rather than cause errors; `termination` steps almost never do; the suspicion distribution over a trace should be sparse). We use the same prompt verbatim on TRAIL and across all three models; the Who&When prompting baselines (Step-by-Step, Binary Search, All-at-Once)

in Table 1 are run on their original prompts (Zhang et al., 2025b) without these priors.

### A.3 Patch generation prompts

Patch generation runs in two LLM calls per candidate: a *hypothesis* call classifies the candidate's error and a *patch* call emits the rewrite. Both share the normalized trace prefix and run at temperature 0. The hypothesis call returns a JSON object with the inferred taxonomy class and a short rationale; its prompt enumerates the operational categories with one-line definitions and includes a category-selection rubric. The residual unknown class in Table 9 is a fallback for unparsable or otherwise unplaceable cases, not a class the prompt encourages the model to select.

**Hypothesis prompt**

```
{task_and_trace_prefix}
## Target step under classification
Target step (Step {target_step_id}): Agent: {agent};
      Role: {role}; Kind: {kind}.
Content:
---BEGIN STEP CONTENT---
{target_content}
---END STEP CONTENT---

ORIGINAL TASK: {question}
# (CORRECT ANSWER field omitted: the groundtruth_answer
      slot is left unset in all pipeline runs; see
      Appendix A header.)

## Your role
You are classifying the error type of Step
      {target_step_id} (the TARGET step) in the failed
      multi-agent trace above. Your job is to classify
      the error at the target step, not to solve the
      original task. Return exactly one JSON object
      matching the schema below. Do NOT answer the
      original task question, even if it is short,
      directive, or easy to solve.

ERROR TAXONOMY:
- faulty_reasoning: Incorrect logical deduction or wrong
      inference from correct premises.
- hallucination: Fabricated facts or claims not supported
      by any prior context.
- planning: Bad strategy, wrong decomposition, constraint
      ignorance, or impossible action.
- tool_selection: Chose the wrong tool or function for
      the task.
- tool_arguments: Correct tool, wrong parameters or
      malformed call.
- tool_interpretation: Misread, ignored, or
      misinterpreted tool output.
- code: Bug in generated code (syntax, logic, algorithm).
- memory: Failed to use available context, contradicted
      prior findings, or forgot results.
- verification: Didn't validate intermediate results or
      accepted clearly wrong output.
- communication: Agent miscommunication, unclear
      delegation, or misunderstood handoff.
- premature_conclusion: Terminated or answered too early
      without sufficient evidence.
- format: Malformed output structure, empty response,
      wrong output schema, or truncated output.

Each category implies a different fix. Choose the
      category that best describes the ROOT CAUSE:
- Information available, correct premises, wrong
      conclusion -> faulty_reasoning
- Agent asserted something that appears nowhere in the
      trace -> hallucination
- The approach/strategy is wrong (not the execution) ->
      planning
- A different tool should have been used entirely ->
      tool_selection
- The right tool was used with wrong parameters ->
      tool_arguments
- Tool returned correct data but agent misinterpreted ->
      tool_interpretation
- The step contains executable code that is incorrect ->
      code
- Info established earlier but not used or contradicted
      -> memory
- Agent should have validated but didn't -> verification
- The message between agents is the problem ->
      communication
- The agent declared done when it wasn't ->
      premature_conclusion
- The output structure is malformed or wrong schema ->
      format

Tie-breaking rules:
- If a step's factual content (numbers, prices, lists) is
      entirely wrong but the framing is correct, prefer
      hallucination or format over faulty_reasoning.
- If the target is a tool_call / code-producing step and
      the bug is in query logic, endpoint choice,
      filters, pagination, or executable logic, prefer
      tool_arguments or code over verification. Reserve
      verification for cases where the action is
      otherwise reasonable but the step fails to check
      completeness or correctness before trusting a
      result.
- If the target is an observed tool_result (e.g.,
      exitcode/stdout output or webpage output), do NOT
      call it `code` unless the executable code itself
      appears in this step. Prefer tool_interpretation
      only when the mistake is in reading the observed
      output; otherwise place the bug on the earlier
      tool/action step.

Respond in this exact JSON format and nothing else:
{
  "taxonomy": "<one of the listed categories>",
  "rationale": "<1-2 sentence explanation of the error>",
  "predicted_local_effect": "<1 sentence: the specific
      immediate consequence if this step were corrected>"
}
```

The patch call receives the normalized trace, the target step in context, the inferred taxonomy, the allowed operator set for that taxonomy class, and (on retry) the prior validator's failure reason. The operator-description block enforces that each operator means a specific kind of local edit:

**Operator descriptions (patch call)**

```
CORRECT_REASONING: Rewrite the logical chain with correct
      inference while preserving grounded premises.
REPLACE_PLAN: Replace the strategy or decomposition only
      when the plan is wrong or missing a concrete
      constraint.
FIX_TOOL_CALL: Fix a local tool, command, function, or
      argument when the call itself is wrong.
FIX_INTERPRETATION: Correct how observed output or tool
      results are interpreted without inventing unseen
      results.
FIX_CODE: Fix executable code (syntax, logic, algorithm,
      imports) while preserving non-code text.
ADD_VERIFICATION: Insert a concrete validation step that
      would catch the local error before proceeding.
INJECT_CONTEXT: Add missing context available earlier in
      the trace, or replace unsupported claims with
      grounded prefix context.
CORRECT_CONCLUSION: Fix the step's bottom-line answer or
      delegation message while preserving grounded
      reasoning.
REPLACE_OUTPUT: Replace substantive content when wrong,
      keeping the repair local and role-consistent.
```

```
ABSTRACT_REPAIR: Produce an honest schematic repair when
      no grounded verbatim patch can be materialized
      without inventing unseen results.
OTHER: Use only if no standard operator fits cleanly. Do
      not use OTHER to justify hindsight editing.
```

The patch call's full constraint block follows. The block is identical across taxonomy classes; only the `OPERATOR GUIDANCE` substitution changes based on the inferred class. The output schema below is parsed by a JSON parser.

**Patch prompt body**

```
## Your role
You are generating a local patch for one step in the
      failed multi-agent trace. Your job is to generate a
      local patch, not to solve the original task. Do NOT
      answer the original task question, even if it
      appears simple or directive.

OPERATOR GUIDANCE:
For error type '{taxonomy}', start with
      {primary_operator} if it fits. Also reasonable:
      {additional_valid_ops}. Use OTHER only if none of
      the listed operators fits the local repair.

CONSTRAINTS:
- Return JSON only; do not return an empty response.
- The patch must stay local to THIS step and preserve as
      much unchanged text as possible.
- Use only information available up to and including this
      step.
- The patch must be role-consistent for the target agent
      and chronologically honest.
- Tool names cannot be changed for FIX_TOOL_CALL unless
      the error type is tool_selection.
- Do NOT patch runner or harness boilerplate (e.g., "No
      agent selected", "SCENARIO.PY COMPLETE").
- Do NOT change the task itself. Preserve entities,
      dates, versions, URLs, requested outputs, and
      numeric constraints unless the trace context
      directly shows they are wrong.
- If you change a number, named entity, URL, or factual
      claim, it must be justified by prefix evidence
      rather than a guess.
- Do NOT invent unseen website contents, tool outputs,
      API responses, execution results, or downstream
      discoveries.
- If the exact corrected content is underdetermined from
      the prefix, make the smallest honest local repair
      and use OTHER if needed.

STEP-SPECIFIC GUIDANCE: (inserted conditionally on
      role/family/kind, e.g.:)
- Do NOT use REPLACE_PLAN unless this step is genuinely
      an explicit strategy or plan block.
- This candidate was screened as a tool/code action.
      Prefer direct local fixes to the call, command, or
      code rather than rewriting surrounding planning
      text.

IMPORTANT:
- Prefer a precise local edit over a full rewrite.
- Do NOT rephrase or rewrite unchanged content. Only
      modify the specific part that is incorrect.
- Keep all other text exactly as-is.
- Your rationale should explain why this patch could
      improve answer recovery without relying on
      hindsight.
```

The output schema is:

**Patch call output schema**

```
{
  "operator": "<one of allowed operators>",
  "target_step_id": <int>,
  "original_content": "<original step text>",
  "patched_content": "<corrected step text>",
  "repair_intent": "<1 sentence>",
  "supporting_evidence": ["<short prefix fact>", ...],
  "rationale": "<brief explanation>"
}
```

### A.4 Replay evaluation prompt

Blinded replay yields two evaluator calls per candidate under one of 10 perspectives (Table 6). Each call presents the partial trace through the checkpoint step (with either the original or the patched step inserted) and asks the evaluator to estimate the success probability under that perspective. The trace prefix is identical across baseline and patched versions; only the step under evaluation differs, and the evaluator never sees both versions simultaneously.

**Replay prompt (perspective variant)**

```
## Task
{question}
# In pipeline runs (Triage attribution, LLM-as-judge
      baseline), the optional groundtruth_answer slot is
      unset, so the evaluator never sees the gold answer;
      see Appendix A header.

## Visible trace
{conversation_through_checkpoint}

## Step under evaluation (Step k)
{step_text}        // original OR patched

## Your role as evaluator
{perspective_preamble}
{perspective_body}

Rate your confidence that the trajectory from this step
      leads to a successful completion of the task above.

Respond with JSON:
{"score": 0.0 to 1.0, "reasoning": "brief explanation"}
```

The repair uplift is $R(i) = 1 + \max(0, \hat{p} - p_0)$, clamped to $\geq 1$ so failed replay never penalizes a candidate (the failure may reflect an incomplete patch or insensitive evaluator rather than a non-causal step).

The ten perspectives are listed in Table 6. Each rotates a different preamble/rubric pair while sharing the perspective-independent prefix (task, visible trace, step under evaluation); in pipeline runs the prefix contains no ground-truth answer. The headline run locks the perspective to id 8 (`critical_error_detection`); the cross-perspective robustness study in Appendix E.8 rotates through all ten.

| id | Perspective | Rubric focus |
|---|---|---|
| 0 | factual_correctness | Is content supported by the prefix or grounded sources? |
| 1 | downstream_traj. | How likely is the trace to recover the correct answer? |
| 2 | task_advancement | Does the step advance the task, or stall / regress? |
| 3 | error_propagation | Does this step's error block all viable continuations? |
| 4 | reasoning_validity | Is the inferential chain valid under available premises? |
| 5 | tool_use_quality | Are tool selection, call signature, and result use appropriate? |
| 6 | info_sufficiency | Is enough grounded information present to proceed? |
| 7 | alt_assessment | Weighs the step against the best plausible alternative. |
| 8 | critical_error_det. | Direct judgment of whether this is the decisive failure (locked default). |
| 9 | holistic_traj_qual. | Whole-trajectory composite of recovery likelihood. |

Table 6: The ten replay perspectives. The headline run locks id 8 (critical_error_detection); Appendix E.8 reports the per-perspective spread.

### A.5 LLM-as-judge baseline prompt

The single-call LLM-as-judge baseline (Figure 2, Table 16) shares the normalized task-and-trace prefix with the suspicion and patch-generation prompts; only the instruction block differs. It asks the LLM in one call to return a top-10 ranked list of decisive-error candidates, the closest single-call analog to TRAIL's exhaustive-enumeration protocol that does not natively yield a top-$K$. Per trace, the target list length is $\min(10, |\tau|)$ where $|\tau|$ is the number of normalized steps, so on traces shorter than 10 steps the LLM must rank all of them.

**LLM-as-judge prompt (instruction block)**

```
## Your role
You are an expert debugger analysing the multi-agent
      conversation above, which FAILED to solve the task.
      Your job is to identify the steps most likely to be
      the decisive error: the step whose correction would
      flip the trace from failure to success.

## What to look for
- A step where an agent makes a factual error, uses a
      wrong tool argument, misinterprets a result, or
      chooses a bad strategy.
- Focus on the EARLIEST mistake that caused later
      failures (root cause), not on downstream symptoms.
- "tool_result" steps are outputs from tools; the mistake
      is usually in the preceding tool_call or reasoning
      step.
- "termination" steps are almost never the root cause.

## Instructions
1. List the 10 steps most likely to be the decisive
      error, ordered from MOST likely (first) to LEAST
      likely (last). Your list MUST contain EXACTLY 10
      step IDs; include weaker candidates at the end of
      the list rather than truncating early.
2. Use only step_ids that appear in the trace above; do
      not repeat IDs.
3. Output the result on the LAST line, in EXACTLY this
      format:
RANKED: [step_id_1, step_id_2, ...]
```

We run the prompt at temperature 0 with the same Kimi K2.5 model used for TRIAGE's suspicion stage. Of 178 evaluable Who&When traces, 175 produced parseable rankings (3 parse failures, counted as misses against $n$=184). On TRAIL, 132 of 148 traces produced parseable rankings (16 misses against $n$=148, concentrated on long GAIA traces where the LLM truncated the response).

## B Structural Impact Graph

### B.1 Anchors and edge sensitivity

For each candidate $i$, we collect a bounded set $A(i) \subseteq \{j : j > i\}$ of downstream anchors selected by structural role: the next plan, tool_call, tool_result, tool_interpretation, factual_assertion, conclusion, and the trajectory's final step. Duplicates are removed and the set is capped at $|A(i)| \leq 6$ to bound graph cost and avoid over-counting in long HC traces; for terminal candidates with no downstream steps, $A(i) = \{i\}$ with a self-anchor flag and $s_{ii} = 1$. For each anchor $j \in A(i)$, edge sensitivity is

$$s_{ij} = \text{clip}_{[0,1]}(d_{ij} \cdot f_{ij} + \beta \cdot \mathbf{1}[\phi(i) = \phi(j)]) \,, \tag{8}$$

where $d_{ij} = 1/(1+\log(1+|j-i|))$ is a logarithmic distance decay, $f_{ij}$ is a step-family pair weight (Table 7), and $\beta = 0.3$ is a same-agent boost (errors propagate more strongly within a single agent's reasoning chain).

### B.2 Criticality and aggregation

Each anchor $j$ receives

$$c_j = \text{clip}_{[0,1.5]}\left(c_j^{\text{type}} \cdot (1 + \max(0, \rho(j)))\right) , \tag{9}$$

where $c_j^{\text{type}}$ is a base weight by step family (Table 8); anchors with high $\rho(j)$ receive amplified criticality so that propagation through a likely-erroneous downstream step counts more than propagation through a clean one, and termination anchors receive criticality 1.0 directly. The aggregated graph score is

$$I_{\text{graph}}(i) = \text{clip}_{[0,1]}\Big(\tfrac{1}{|A(i)|} \sum_{j \in A(i)} s_{ij} \cdot c_j\Big),$$

mixed with the position prior as $I(i) = \alpha I_{\text{graph}}(i) + (1-\alpha)/(1+i)$. All constants and weights below were set by hand on Who&When and held fixed for TRAIL.

| target \ anchor | T-act | T-int | Plan | Concl. | Fact. |
|---|---|---|---|---|---|
| Tool-action | N/A | 0.9 | N/A | 0.6 | 0.5 |
| Tool-interp. | 0.5 | N/A | N/A | 0.7 | 0.6 |
| Plan | 0.8 | 0.5 | N/A | 0.6 | 0.5 |
| Factual | 0.5 | N/A | N/A | 0.7 | 0.5 |
| Delegation | 0.6 | N/A | N/A | 0.5 | 0.6 |
| Other | N/A | N/A | N/A | 0.5 | 0.5 |

Table 7: Step-family pair weights $f_{ij}$ for selected anchor types. Unlisted pairs default to 0.4; N/A marks inapplicable self-family anchors. T-act = tool action, T-int = tool interpretation.

| Family | $c^{\text{type}}$ | Family | $c^{\text{type}}$ |
|---|---|---|---|
| Conclusion | 0.9 | Tool-action | 0.6 |
| Tool-interp. | 0.7 | Factual | 0.6 |
| Plan | 0.7 | Delegation | 0.5 |
| Other / Termination* | | | 0.5 / 1.0 |

Table 8: Per-family base criticality weights $c^{\text{type}}$. Termination anchors receive a flat 1.0 regardless of inferred family.

Other fixed hyperparameters: same-agent boost $\beta = 0.3$, mix coefficient $\alpha = 0.7$, anchor cap $|A(i)| \leq 6$, suspicion floor 0.01, $P = 5$ suspicion passes, top-$M = 3$ for replay.

# C Repair Generation

## C.1 Error taxonomy and patch operators

TRIAGE's patch generation conditions on a 13-class error taxonomy. The taxonomy was hand-designed to cover the failure modes observed in MAST-Data (Cemri et al., 2026) and Who&When traces, with one residual category (`unknown`) for steps the classifier cannot place. Each taxonomy class maps to one *primary* patch operator (the default rewrite shape for that error type) and zero or more *additional valid* operators that the patch call may select if the primary does not fit the local repair (Table 9). The hypothesis call selects the taxonomy class; the patch call is constrained to the allowed operator set for that class plus `OTHER`. This separation prevents the patch model from using hindsight to rewrite the step: it must commit to an error type first and then make a local edit consistent with that type.

Table 10 shows the empirical class and operator distribution over the 530 patches generated for the headline Who&When run.

| Error category | Primary operator (also valid) |
|---|---|
| Faulty reasoning | Correct reasoning (Correct conclusion) |
| Hallucination | Inject context (Correct reasoning, Replace output) |
| Planning failure | Replace plan (Correct reasoning) |
| Tool selection | Fix tool call (Replace plan) |
| Tool arguments | Fix tool call |
| Tool interpretation | Fix interpretation (Correct conclusion) |
| Code error | Fix code |
| Memory / context | Inject context (Correct reasoning) |
| Missing verification | Add verification (Inject context) |
| Communication error | Correct conclusion (Inject context) |
| Premature conclusion | Correct conclusion (Add verification) |
| Format error | Replace output (Correct conclusion) |
| Unknown | Correct reasoning |

Table 9: 13 error categories with primary patch operator and additional valid operators. The hypothesis call infers the category; the patch call selects from the allowed operator set.

| Taxonomy class | $N$ | Operator | $N$ |
|---|---|---|---|
| `planning` | 132 | `CORRECT_REASONING` | 115 |
| `hallucination` | 95 | `ABSTRACT_REPAIR` | 109 |
| `code` | 59 | `REPLACE_OUTPUT` | 62 |
| `faulty_reasoning` | 55 | `REPLACE_PLAN` | 53 |
| `tool_interpretation` | 39 | `FIX_CODE` | 52 |
| `tool_arguments` | 33 | `FIX_TOOL_CALL` | 34 |
| `verification` | 31 | `CORRECT_CONCLUSION` | 29 |
| `premature_conclusion` | 28 | `FIX_INTERPRETATION` | 28 |
| `tool_selection` | 20 | `INJECT_CONTEXT` | 25 |
| `communication` | 18 | `ADD_VERIFICATION` | 15 |
| `memory` | 11 | `OTHER` | 8 |
| `format` | 9 | | |
| Total | 530 | Total | 530 |

Table 10: Hypothesis-class and operator distributions over 530 patches (178-trace Who&When run). `ABSTRACT_REPAIR` is the rule-based fall-back.

## C.2 Grounded vs. abstract repairs

A grounded repair rewrites the step's content with a minimal, near-verbatim fix (e.g., a one-token correction in a code block, a corrected pronoun in a translation). Some steps, most often web-browsing observations whose content depends on live search results, cannot be faithfully rewritten without fabricating unseen output. For these, we fall back to an *abstract repair*: a schematic description of the intended change (e.g., "replace the search-result URL with the DOI link"). Abstract repairs are flagged in the candidate record so downstream evaluation knows they are sketches. The decision is rule-based, conditioning on step kind, code-likeness, and observed-tool-result status; if none of the grounded operators applies, the step falls back to `ABSTRACT_REPAIR`.

## C.3 Patch checks and fallback rules

Generated patches are checked against per-operator Levenshtein-ratio caps (Table 11); patches that

change too much of the original step (above the cap) are rejected and re-generated with a feedback message. Up to two retries are attempted per candidate. Validators also check role consistency (an Orchestrator step must remain a delegation or plan; a tool-call step must remain a tool-call), JSON-schema conformance for structured steps, and that the patch does not invent unseen tool results. Patches failing all retries are marked *unpatched*, are not replayed, and contribute $R(i) = 1$.

| Operator | Cap | Operator | Cap |
|---|---|---|---|
| Correct reasoning | 0.65 | Add verification | 0.60 |
| Replace plan | 0.75 | Inject context | 0.60 |
| Fix tool call | 0.60 | Correct conclusion | 0.60 |
| Fix interpretation | 0.65 | Replace output | 0.70 |
| Fix code | 0.75 | Abstract repair | 1.00 |

Table 11: Per-operator Levenshtein-ratio caps.

### C.4 Promotion-only replay design

Section 4 defines the replay uplift as $R(i) = 1 + \max(0, \hat{p}(i) - p_0(i))$, with $R \in [1, 2]$ for replayed candidates and $R = 1$ for non-replayed candidates. The clamp at $R \geq 1$, which prevents failed replay from pulling a candidate's score below its $\rho \cdot I$ baseline, reflects three observations.

**Failed replay is ambiguous.** A patched trajectory $\hat{p}$ may fail to exceed the baseline $p_0$ for at least three reasons: the step is not actually the decisive error, the generated patch is incomplete or addresses only a symptom, or the LLM evaluator is insensitive to the change. Only the first reflects negative evidence about step causality; the other two are evidence about patch quality or the evaluator. Treating $R < 1$ as a demotion signal would conflate these regimes and systematically penalize candidates whose patches happened to be incomplete.

**Patches and evaluators are imperfect.** $\sim$21% of TRIAGE's repairs are abstract fallbacks (Section C, e.g., "use DOI link instead of search") rather than grounded rewrites; abstract repairs cannot be replayed faithfully. Even on grounded repairs, replay does not always return $\hat{p} > p_0$ on the causal step. Treating absence of evidence as negative evidence would systematically penalize correct attribution.

**Empirical support.** The formula ablation (Appendix E.6) shows that adding $R$ to $\rho \cdot I$ changes top-1 from 44.3% to 43.8% (within sampling noise) while improving NDCG@3 from .575 to .579; replay refines ordering without hurting localization, the design intent. The full-replay variant, which scores every candidate without the floor, drops top-1 to 39.8% (4.0pp below the cascade-with-floor) despite using 6$\times$ more replay calls. The promotion-only floor is therefore both conceptually motivated and essentially free: $\rho \cdot I$ provides the floor ranking, and $R$ adds evidence on the top-$M$ candidates that pass the cascade gate.

## D TriageBench Construction

### D.1 Candidate pool construction

For each of 177 Who&When traces (both AG and HC subsets), we form a candidate pool of up to 8 steps from the normalized trace, always retaining the human-annotated decisive-error step (mean 6.7 candidates per trace, 1,184 total). Pool size per trace scales with trace length; the pool is method-agnostic, depending on no attribution system's signals.

### D.2 Pairwise annotation and Bradley-Terry aggregation

Each council member sees the trace and the ground-truth answer. For each ordered candidate pair (A, B), the annotator returns one of six verdicts: `A_strong`, `A`, `tie`, `B`, `B_strong`, `insufficient_evidence`, together with a per-side *answer-recovery* estimate (high / medium / low / none) for A and B. Candidate identifiers are randomly relabeled per pair to prevent position bias, and a `bias_check` block records whether step position, length, or type influenced the choice. Pairwise votes are aggregated via the Davidson model (Davidson, 1970), a Bradley-Terry extension with an explicit tie parameter $\nu$. We fit a Bayesian variant with weakly-informative priors over per-candidate strength parameters $\theta_k$ and the tie scale ($\theta_k \sim \mathcal{N}(0, 1)$, $\log \nu \sim \mathcal{N}(0, 1)$), drawing 5,000 posterior samples per trace and reporting the posterior mean as the consensus severity.

### D.3 Council prompt

Each council member (GPT-5.4, Claude Opus 4.6, Grok 4.20) receives the trace, the candidate pool, the ground-truth expected answer, and the pairwise instruction below. The criterion is which local repair would most plausibly increase the probability of recovering the correct final answer; the prompt explicitly bans ranking by root cause, earliest error,

or position. A required `bias_check` block asks the annotator to declare whether step position, length, or type influenced the verdict.

**Council prompt (excerpt)**

```
You are annotating a repair-priority benchmark for failed
      multi-agent traces.

Your job is not to identify the root cause, assign blame,
      or guess what the original system predicted. Your
      job is to judge which single local repair would
      most plausibly increase the probability of
      recovering the known correct answer.

## Pairwise scale (5 points + abstain)
- A_strong: A clearly stronger by expected answer-recovery
- A: A somewhat stronger
- tie: similar expected effect (including both near zero)
- B: B somewhat stronger
- B_strong: B clearly stronger
- insufficient_evidence: packet does not support a
      reliable comparison

## Important rules
- Do not infer from hidden metadata.
- Do not rank by root cause, blame, or earliest error.
- Do not reward earlier position unless it clearly
      changes more of the future trajectory.
- Do not let step length or verbosity influence the
      judgment.
- Do not automatically down-rank routing, delegation, or
      orchestration steps.
- Root cause and best repair target may differ.
- In bias_check, explicitly say whether step position,
      length, or type affected your choice.

## Output (JSON)
Return one JSON object only:
{
  "winner": "A_strong | A | tie | B | B_strong |
      insufficient_evidence",
  "answer_recovery": {"A": "high | medium | low | none",
      "B": "high | medium | low | none"},
  "rationale": "<2-3 sentences focusing on expected
      answer recovery under each repair>",
  "bias_check": {
    "positional_bias": "yes | no",
    "length_bias": "yes | no",
    "type_bias": "yes | no",
    "note": "<1 sentence; required if any bias is yes>"
  }
}
```

Pairs are sampled by stratifying on $\rho$ deciles and step kind to balance the council workload across easy/hard and structurally varied candidates. The visible packet contains no metadata about which model generated which candidate or about prior verdicts on adjacent pairs.

### D.4 Agreement and posterior diagnostics

Diagnostics (Table 12) are computed over the supplementary TriageBench traces ($n$=177 traces, mean 6.7 candidates per trace; 3,629 ordered pairwise votes). Pairwise winner agreement reaches Fleiss $\kappa = 0.646$ (substantial) and Krippendorff $\alpha = 0.594$; the 5-point ordinal $\alpha$ is 0.676. These levels are comparable to or exceed the inter-annotator agreement reported on Who&When's human annotations (9.1 to 50% step-level disagreement). The mean number of `insufficient_evidence` votes per pair (over 3 council members) is 0.22, indicating most pairs support a confident judgment. For a representative trace where the Orchestrator commits prematurely to a faulty translation, the council unanimously picks the conclusion-rewrite candidate over a plan-rewrite candidate (`A_strong`/`A_strong`/`A`); recovery anchors are high/high/medium; bias-check declares no positional influence, and the posterior gives the conclusion-rewrite candidate strength 0.78 versus 0.14 for the plan-rewrite candidate.

| Quantity | Value |
|---|---|
| *Per-trace Bradley-Terry posterior* | |
| BT-score variance, mean / median | 1.221 / 1.248 |
| BT-score variance, p25 / p75 | 0.845 / 1.524 |
| BT spread (max−min), mean / med. | 2.948 / 3.049 |
| BT spread, p25 / p75 | 2.370 / 3.561 |
| Top-1 posterior prob., mean | 0.795 |
| Top-1 posterior prob., median | 0.837 |
| Traces with top-1 post. $> 0.50$ | 158 / 170 (92.9%) |
| Traces with top-1 post. $> 0.70$ | 118 / 170 (69.4%) |
| Traces with top-1 post. $> 0.90$ | 68 / 170 (40.0%) |
| *Three-way council pairwise verdicts* | |
| Cohen $\kappa$, GPT vs. Opus (3-cls) | 0.638 |
| Cohen $\kappa$, GPT vs. Grok (3-cls) | 0.531 |
| Cohen $\kappa$, Opus vs. Grok (3-cls) | 0.599 |
| QWK, GPT vs. Opus (5-pt) | 0.721 |
| QWK, GPT vs. Grok (5-pt) | 0.625 |
| QWK, Opus vs. Grok (5-pt) | 0.690 |
| Fleiss $\kappa$, 3 raters (3-cls) | 0.646 |
| Three-way unanimous (complete cases) | 1,228 / 1,658 (74.1%) |

Table 12: TriageBench posterior and inter-annotator diagnostics. Council: GPT-5.4, Claude Opus 4.6, Grok 4.20. Posterior rows use $n$=170 (traces with $\geq 2$ candidates after BT fit; 7 of 177 excluded). Complete-case rows restrict to pairs where all three annotators voted. QWK = quadratic-weighted Cohen $\kappa$ on the 5-point pairwise scale.

The BT posterior is concentrated: 92.9% of traces have a single candidate carrying $> 50\%$ of top-1 mass, and 40% have $> 90\%$ on a single step, ruling out the concern that the council was forced into near-uniform ties. Quadratic-weighted 5-point $\kappa$ values (0.625 to 0.721) show that disagreements concentrate on adjacent strength levels (`A` vs. `A_strong`) rather than on which candidate wins. Pairwise winner agreement is strongest between GPT-5.4 and Claude Opus 4.6 ($\kappa$=0.638) and lowest with Grok 4.20.

**Per-LLM agreement with the human gold.** Inferring each council member's top-1 from its own pairwise votes (most pairwise wins), per-LLM agreement with the Who&When human-annotated decisive step is 29.9% (GPT-5.4), 44.6% (Claude Opus 4.6), and 41.5% (Grok 4.20); the BT-aggregated council top-1 reaches 40.7%. Even

though the three LLMs cast unanimous winners on 74.1% of 3-way pairs, they collectively disagree with the human gold on ∼60% of traces. The agent-level agreement is uniformly higher: the council BT top-1 agent matches the human-annotated faulty agent in 58.2% of traces, +18pp over step-level. Where the council disagrees on the step, it picks a later step than the human in 59% of disagreements (mean offset +2.14 steps, median +1), shifting the normalized trace position from human-median 0.30 to council-median 0.40. Plausibility "yes" rates per annotator are 65.6% (GPT-5.4), 72.9% (Opus 4.6), and 62.5% (Grok 4.20), so roughly two-thirds of the randomly sampled candidates are judged viable repair targets by each LLM, consistent with the view that traces contain multiple interacting errors. Admissibility labels reinforce this: 174/177 traces (98.3%) are tagged "single-step intervention priority but not a clean decisive error"; only 3 are "single-step decisive error."

**Council vs. human top-$K$ coverage.** Although the council's BT top-1 matches the human-annotated step in only 40.7% of traces, the human step appears within the council's top-3 in 68.9%, top-5 in 84.2%, and top-10 in 100%, mirroring the same single-step vs. ranked-coverage pattern observed for TRIAGE on Who&When (Section 6.2). The candidate pool itself contains the human gold step in 177/177 traces, so pool selection is not the bottleneck.

# E Additional Experiments

## E.1 Evaluable subset

We evaluate on Who&When and TRAIL with the candidate pool set to every normalized trace step. Eight Who&When traces are excluded from formula-level analysis: 6 fail normalization validation as `mistake_agent_mismatch` (the step at the annotated `mistake_step` index is performed by an agent other than the annotated `mistake_agent`, leaving the decisive step undefined), and 2 exceed Kimi K2.5's 128K context budget. The six affected UUIDs (3 AG, 3 HC) are `1dcc160f`, `dc22a632`, `851e570a` (Algorithm-Generated) and `5f982798`, `14569e28`, `b816bfce` (Hand-Crafted); Zhu et al. (2026) independently report the same agent-name mismatch in their Who&When analysis. The remaining $n$=176 traces form the *evaluable subset*, used in the formula ablations (Tables 18, 14) because including unscorable traces would conflate ranking quality with annotation noise. Main-body comparisons against prior methods (Table 1, Figure 2) use the full $n$=184 denominator and count all 8 dropped traces as misses, matching the denominator used by prior work. On TRAIL, all 148 traces are evaluable; SWE-Bench labels whose annotated spans straddle multiple steps are credited as a hit if any step in the span is recovered.

## E.2 Random and formula baselines

Two questions motivate baseline comparisons: (i) does producing a ranked list at all help (random vs. TRIAGE top-$K$, Table 13)? and (ii) does TRIAGE's ranking formula help relative to simpler ranking functions over the same pool (Table 14)? We treat them separately because they answer different questions.

**Random top-$K$ (Table 13).** The random baseline samples $K$ steps uniformly without replacement from the entire normalized trace, the same pool TRIAGE ranks over, and counts a hit if any sample matches an annotated error. We use Monte Carlo with 200 seeds per trace.

| $K$ | Random | TRIAGE | Δ |
|---|---|---|---|
| *Who&When ($n$=184, single error per trace)* | | | |
| 1 | 9.2 | **42.4** | +33.2 |
| 3 | 28.0 | **64.1** | +36.1 |
| 5 | 46.4 | **73.9** | +27.5 |
| 10 | 77.7 | **83.2** | +5.5 |
| *TRAIL ($n$=148, ∼5.7 errors per trace)* | | | |
| 1 | 14.0 | **56.8** | +42.8 |
| 3 | 37.5 | **77.7** | +40.2 |
| 5 | 55.0 | **84.5** | +29.5 |
| 10 | 82.6 | **93.2** | +10.6 |

Table 13: Random top-$K$ vs. TRIAGE (Kimi K2.5) recall (%). Random samples $K$ steps uniformly from the normalized trace.

| Baseline | Top-1 | Within-2 | MRR | NDCG@5 |
|---|---|---|---|---|
| Position $\pi$ only | 12.0 | 35.9 | .326 | .368 |
| Suspicion $\rho$ only | 43.8 | 64.2 | .576 | **.588** |
| TRIAGE ($\rho I$) | **44.3** | **64.8** | **.586** | .583 |

Table 14: Ranking-formula comparison on the Who&When evaluable subset ($n$=176).

**Formula comparison on a fixed pool (Table 14).** On the same evaluable subset as Table 18, we compare TRIAGE against simpler ranking functions: position-only ($\pi(i) = 1/(1+i)$, no LLM) and suspicion-only ($\rho$). Position-only reaches 12.0%

top-1. Suspicion-only ($\rho$) reaches 43.8%, within 0.5pp of TRIAGE's 44.3%, consistent with the finding in Section 6.3 that $\rho$ carries most of TRIAGE's localization signal.

### E.3 Multi-model LLM-as-judge

We ran the LLM-as-judge prompt (Appendix A.5) on three models to test whether the cascade's top-$K$ advantage holds beyond Kimi K2.5 (Table 15). Two findings: (i) the single-call cap is structural across models, with all three LLM-as-judge configurations saturating in the ∼80 to 85% range at $K$=10, consistent with emitting at most 10 candidates from a ∼22-step pool; (ii) TRIAGE on Kimi K2.5 leads at $K$=1 and remains competitive at higher $K$, while stronger LLM-as-judge configurations (Sonnet 4 at $K$=3−5, Llama 3.3 70B at $K$=10) close the top-$K$ gap. The cascade's value over enumeration is therefore strongest at $K$=1 and in producing repair patches and uplift estimates, rather than uniformly dominating every Who&When top-$K$ point.

| Model (target = 10) | $K$=1 | $K$=3 | $K$=5 | $K$=10 |
|---|---|---|---|---|
| Kimi K2.5 | 39.7 | 63.0 | 72.3 | 79.9 |
| Claude Sonnet 4 | 41.3 | **66.3** | **78.8** | 83.2 |
| Llama 3.3 70B | 30.4 | 53.3 | 71.7 | **84.8** |
| TRIAGE (Kimi) | **42.4** | 64.1 | 73.9 | 83.2 |

Table 15: Multi-model top-$K$ recall (%) on Who&When ($n$=184) for LLM-as-judge target = 10 vs. TRIAGE. The cascade leads at $K$=1; stronger LLM-as-judge configurations (Sonnet 4 at $K$=3−5, Llama 3.3 70B at $K$=10) match or edge TRIAGE at higher $K$.

### E.4 $\rho$-only suspicion ablation

The $\rho$-only row in Table 16 runs TRIAGE's suspicion scoring prompt at $P$=5 with mean aggregation (variance reduction over the same single-call all-steps prompt), ranks the full step pool by suspicion score, and reports top-$K$ recall. This is the cascade's suspicion stage in isolation: the single-call reformulation without $I$ or $R$. The gap from $\rho$-only to full TRIAGE ($\rho IR$) isolates the cascade's ranking contribution (Section 6.3): up to +0.6pp on Who&When and +2.1pp on TRAIL at top-1/top-10, near-zero at intermediate $K$. Twelve of 332 traces hit the 600s timeout on the longest TRAIL/SWE-Bench contexts even at `max_tokens`= 32768 and are counted as misses in the denominators above.

| Method | $K$=1 | $K$=3 | $K$=5 | $K$=10 |
|---|---|---|---|---|
| *Who&When ($n$=184)* | | | | |
| LLM-judge (target 10) | 39.7 | 63.0 | 72.3 | 79.9 |
| $\rho$-only | 41.8 | **65.2** | **74.5** | 82.6 |
| TRIAGE ($\rho IR$, $P$=5) | **42.4** | 64.1 | 73.9 | **83.2** |
| *TRAIL ($n$=148, any-error hit)* | | | | |
| LLM-judge (target 10) | 52.0 | 70.3 | 75.0 | 80.4 |
| $\rho$-only | 54.7 | **78.4** | 84.5 | 91.2 |
| TRIAGE ($\rho IR$, $P$=5) | **56.8** | 77.7 | 84.5 | **93.2** |

Table 16: LLM-as-judge vs. $\rho$-only vs. TRIAGE on Who&When and TRAIL top-$K$ recall (%). target = 10 is a single-call ranked list capped at 10; $\rho$-only is TRIAGE's suspicion stage in isolation.

### E.5 Paired bootstrap confidence intervals

For each (dataset, $K$) comparison, we draw 10,000 paired bootstrap resamples (same indices for both methods) and report the mean delta with 95% percentile CI and two-sided $p$-value. Table 17 compares TRIAGE's full ranking against the target = 10 LLM-as-judge baseline at the candidate-covered paired subsets ($n$=175 W&W, $n$=144 TRAIL); these paired deltas are slightly larger than the n=184/n=148 aggregate deltas in the main body because the paired denominator drops traces where one method failed. On TRAIL, TRIAGE's top-3/5/10 advantages are statistically significant ($p$=0.049/0.009/< 0.001); top-1 is marginal ($p$=0.169). On Who&When, no $K$ shows a statistically significant gap ($p$>0.08 across all $K$), supporting the saturation reading: top-1 plateaus near the annotation-disagreement floor and top-$K$ saturates by $K$=10.

| $K$ | Δ (pp) | 95% CI | $p$ |
|---|---|---|---|
| *Who&When ($n$=175 paired, single-error hit)* | | | |
| 1 | +1.7 | [−3.4, +6.9] | 0.604 |
| 3 | +2.9 | [−2.3, +8.0] | 0.318 |
| 5 | +4.6 | [+0.0, +9.7] | 0.083 |
| 10 | +1.1 | [−2.3, +4.6] | 0.639 |
| *TRAIL ($n$=144, any-error hit)* | | | |
| 1 | +4.9 | [−1.4, +11.1] | 0.169 |
| 3 | +7.6 | [+0.7, +14.6] | **0.049** |
| 5 | +9.7 | [+2.8, +17.4] | **0.009** |
| 10 | +13.2 | [+6.9, +19.4] | **<0.001** |

Table 17: Paired bootstrap (10K resamples) for TRIAGE minus target = 10 LLM-as-judge top-$K$ recall. Positive Δ favors TRIAGE. CIs are percentile (2.5th, 97.5th); $p$-values are two-sided.

### E.6 Formula ablation

Table 18 reports the full formula ablation on the Who&When evaluable subset ($n$=176). All rows rerank the same candidates and replay outputs; only the ranking formula changes. TRIAGE's top-1 here (44.3%) differs from Table 1 (42.4%) because the latter uses the full $n$=184 denominator and counts the 8 dropped traces as misses.

| Formula | T1 | W2 | T5 | MRR | N@3 |
|---|---|---|---|---|---|
| Position ($\pi$) | 11.9 | 36.9 | 59.7 | .336 | .282 |
| Suspicion ($\rho$) | 43.8 | 64.2 | **77.8** | .576 | .574 |
| $\pi \cdot \rho$ | 39.8 | 61.4 | 76.1 | .563 | .567 |
| $\rho \cdot I_{\text{graph}}$ | 43.2 | 63.6 | 75.0 | .577 | .561 |
| $\rho \cdot I$ | **44.3** | **64.8** | 77.3 | **.586** | .575 |
| $\rho I R$ (TRIAGE) | 43.8 | 64.2 | **77.8** | .585 | **.579** |
| Full-replay $\pi\rho\lambda$ | 39.8 | 60.8 | 77.3 | .566 | .570 |
| Learned (LOOCV) | 42.4 | N/A | N/A | N/A | N/A |

Table 18: Formula ablation on the Who&When evaluable subset ($n$=176). T1, W2, T5, N@3 = Top-1, W±2, Top-5, NDCG@3. Best per column **bold**; second-best underlined.

$\rho$-only top-5 (77.8%) marginally exceeds $\rho \cdot I$ top-5 (77.3%); the gap is within ablation noise, and the top-K advantage of $\rho \cdot I$ over $\rho$-only concentrates in W±2 (+0.6pp) and MRR (+0.010), with NDCG@3 essentially tied. Adding $R$ does not improve top-1 (44.3% to 43.8%): $R \geq 1$ is a multiplicative promotion-only factor applied only to the top-$M$ candidates already ranked by $\rho \cdot I$, so replay cannot rescue a low-$\rho$ candidate. We retain $R$ for actionability (each surviving candidate receives a generated repair patch and a per-candidate uplift estimate $\hat{p} - p_0$); the cascaded design issues ∼80% fewer replay calls than full-replay (534 vs. ∼3,000 projected on Who&When) while scoring +4.0pp higher top-1.

| Formula | T1 | T3 | T5 | T10 | MRR |
|---|---|---|---|---|---|
| Position ($\pi$) | 0.7 | 67.4 | 75.7 | 91.0 | .341 |
| Suspicion ($\rho$) | 56.2 | **80.6** | 86.8 | 93.8 | .694 |
| $\pi \cdot \rho$ | 52.8 | 79.2 | 87.5 | 91.0 | .671 |
| $\rho \cdot I_{\text{graph}}$ | **59.0** | 79.2 | **88.2** | **96.5** | **.712** |
| $\rho \cdot I$ (TRIAGE) | 58.3 | 79.9 | 86.8 | 95.8 | .708 |
| Full-replay $\pi\rho\lambda$ | 53.5 | **80.6** | 87.5 | 91.7 | .676 |

Table 19: Formula ablation on the TRAIL evaluable subset ($n$=144, any-error hit). Best per column **bold**; second-best underlined.

Table 19 extends the ablation to TRAIL ($n$=144 evaluable subset). On multi-error TRAIL, $\rho \cdot I_{\text{graph}}$ (no position mix) edges $\rho \cdot I_{\text{mix}}$ (TRIAGE headline) by +0.7pp at $K$=1 and $K$=10; we retain the mix for cross-dataset consistency with Who&When, where mixing in $\pi$ adds +1.1pp top-1. The body reports these numbers at the full $n$=148 denominator (counting four timeout/parse failures as misses): 54.7/78.4/84.5/91.2 for $\rho$-only, 56.8/77.7/84.5/93.2 for TRIAGE.

| Variant | T1 | T3 | T5 | T10 | W±2 |
|---|---|---|---|---|---|
| $\rho$-only | 43.3 | **67.4** | 77.0 | 85.4 | 63.5 |
| $\rho \cdot I$ (uniform) | 42.7 | 66.9 | **77.5** | **86.0** | 62.9 |
| $\rho \cdot I$ (hand-tuned) | **43.8** | 66.3 | 76.4 | **86.0** | **64.0** |

Table 20: Uniform-weight ablation on Who&When ($n$=178). T$K$ = Top-$K$ step accuracy (%); W±2 = within-2 accuracy.

**Uniform-weight ablation.** We recompute $I_{\text{graph}}$ on Who&When ($n$=178) with the family-pair weights ($f_{ij}$, Table 7) and per-family criticality ($c^{\text{type}}$, Table 8) collapsed to 0.5, the same-agent boost set to 0, and termination criticality set to 0.5 (anchor selection and distance decay unchanged). Hand-tuned weights gain +1.1pp T1 over uniform (Table 20); the structural-impact contribution comes from the type-weighted aggregation, not the graph structure alone. Absolute gains are small ($\leq$ 1.1pp) and within sampling noise on $n$=178.

| Model | T1 | AG | HC | Top-3 |
|---|---|---|---|---|
| Kimi K2.5 | **42.4** | **52.4** | **20.7** | 63.6 |
| Claude Sonnet 4 | 37.5 | 49.2 | 12.1 | **70.1** |
| Llama 3.3 70B | 29.3 | 36.5 | 13.8 | 45.1 |

Table 21: Cross-model comparison on Who&When ($n$=184). *Top-3*: fraction of traces with the gold step in TRIAGE's top-3 ranking.

### E.7 Additional model results

Beyond the headline Kimi K2.5 model, we ran the full pipeline with Claude Sonnet 4 (RAFFLES's model) and Llama 3.3 70B on Who&When. Table 21 compares the three.

The Sonnet 4 vs. Kimi K2.5 comparison reveals complementary strengths. Sonnet 4's suspicion scores cast a wider net (gold step in cascade-gated top-$M$=3 on 70.1% of traces vs. Kimi's 63.6%), but Kimi's ranking within the $M$=3 pool is sharper (rank-1 accuracy when gold is present: 62.4% vs. Sonnet's 47.3%), giving Kimi the better top-1 (42.4% vs. 37.5%).

A natural extension is a hybrid pipeline that uses Sonnet 4 for the suspicion stage (to maximize gold

recall into top-$M$) and Kimi K2.5 for the final ranking within those candidates. The product upper bound is $0.701 \times 0.624 \approx 43.8\%$ top-1, $+1.3$pp above the best single-model headline. Whether the per-model recall and precision properties transfer when models are swapped mid-pipeline is an open empirical question; we leave the hybrid run as future work.

On the same model as RAFFLES, TRIAGE produces a top-3 ranked list that recalls the gold step in 70.1% of traces and surfaces three ranked repair candidates per trace, an output that single-step methods cannot provide. The cascade savings ($\sim$80% replay reduction, 534 calls) hold across all three models.

### E.8 Robustness across replay perspectives

Replay can be conditioned on any of 10 evaluation perspectives, each probing a different aspect of trace quality. All main-body results use a single perspective (*critical error detection*). Table 22 reports step accuracy across all 10 perspectives on Who&When: the spread is 1.6pp (37.5 to 39.1%), all 10 agree on the top-1 prediction in 91% of traces, and the top-3 set is even more stable (98% agreement). Suspicion averaging over $P = 5$ passes reduces per-step $\sigma$ from $\sim$0.12 to $\sim$0.07, making the ranking reproducible across runs.

| id | Perspective | T1 | W±2 | agree |
|---|---|---|---|---|
| 0 | factual_correctness | 37.5 | 57.6 | 0.96 |
| 1 | downstream_trajectory | 38.0 | 58.2 | 0.96 |
| 2 | task_advancement | 38.0 | 57.6 | 0.96 |
| 3 | error_propagation | 39.1 | 58.7 | 0.98 |
| 4 | reasoning_validity | 38.6 | 58.2 | 0.97 |
| 5 | tool_use_quality | 38.6 | 59.2 | 0.97 |
| 6 | information_sufficiency | 38.6 | 59.2 | 0.96 |
| 7 | alternative_assessment | 38.0 | 58.7 | 0.96 |
| **8** | **critical_error_detection** | **39.1** | **59.8** | **1.00** |
| 9 | holistic_trajectory_quality | 38.0 | 58.7 | 0.98 |
| spread | (max − min) | 1.6 | 2.2 | N/A |

Table 22: Per-perspective replay on Who&When ($n$=184). *agree*: fraction whose top-1 matches the default (id 8).

### E.9 Per-error-category recall on TRAIL

Across the 144 TRAIL traces with gold annotations, the canonicalized labels contain 834 annotated error spans (563 unique after de-duplication on (trace, category)). Top-$K$ recall is the fraction of (trace, category) pairs for which any step in that category is in TRIAGE's top-$K$ ranking; categories are normalized for capitalization and pluralization variants. Table 23 reports categories with $\geq 5$ annotated (trace, category) pairs.

| TRAIL error category | $N$ | Top-1 | Top-3 | Top-5 |
|---|---|---|---|---|
| Language-only | 48 | **58.3** | **87.5** | **89.6** |
| Goal Deviation | 63 | 55.6 | 77.8 | 84.1 |
| Tool-related | 39 | 51.3 | 64.1 | 71.8 |
| Tool Selection Errors | 36 | 50.0 | 91.7 | 94.4 |
| Task Orchestration | 37 | 40.5 | 75.7 | 81.1 |
| Incorrect Problem Ident. | 19 | 36.8 | 68.4 | 84.2 |
| Tool Output Misinterpret. | 14 | 35.7 | 57.1 | 71.4 |
| Environment Setup Errors | 9 | 22.2 | 66.7 | 66.7 |
| Authentication Errors | 5 | 20.0 | 20.0 | 20.0 |
| Instruction Non-compliance | 78 | 16.7 | 51.3 | 75.6 |
| Context Handling Failures | 40 | 12.5 | 15.0 | 20.0 |
| Poor Information Retrieval | 28 | 7.1 | 10.7 | 25.0 |
| Formatting Errors | 73 | 5.5 | 39.7 | 57.5 |
| Resource Abuse | 42 | 2.4 | 26.2 | 35.7 |
| All ($N \geq 5$) | 531 | 29.4 | 55.4 | 66.3 |

Table 23: Top-$K$ recall (%) on TRAIL by error category. TRIAGE preferentially surfaces semantic and propagating errors (Goal Deviation, Tool Selection, Context Handling) over surface-level annotation classes (Formatting, Resource Abuse).

## F Cost Decomposition

Table 24 decomposes the canonical Kimi K2.5 Who&When run by pipeline stage. Each trace issues 1 suspicion call (sampled $P$=5 times for variance reduction in our runs), 6 patch requests ($M$=3 top candidates $\times$ hypothesis-then-patch), and 6 replay requests, totaling 13 LLM calls (17 with the $P$=5 sampling). Patch generation dominates at $\sim$30% of total cost because each call includes the full trace context. Cascading replay to top-$M$=3 rather than the full pool reduces replay calls from $\sim$3,000 (projected) to 534 (observed), an 80% reduction (Section 6.3).

| Stage | Calls | In tok | Out tok | % cost |
|---|---|---|---|---|
| Normalize | 0 | 0 | 0 | 0.0 |
| Suspicion | 890 | 7.7M$^\dagger$ | 0.89M$^\dagger$ | 32.9$^\dagger$ |
| Impact graph | 0 | 0 | 0 | 0.0 |
| Patch hypoth. | 531 | 4.7M | 0.08M | 18.6 |
| Patch gen. | 871 | 7.0M | 0.64M | **29.7** |
| Baseline replay | 534 | 1.9M | 0.08M | 8.3 |
| Patched replay | 497 | 1.7M | 0.08M | 7.7 |
| Total | 3,323 | $\sim$23.0M | $\sim$1.8M | 100 |

Table 24: Cost decomposition for the Kimi K2.5 Who&When run ($n$=178, $\sim$\$14.2 total). $^\dagger$Suspicion estimated from per-call average (logs not persisted).

## G Human + LLM-Panel Patch Rating Study

**Motivation.** The replay uplift estimates ($\hat{p} - p_0$) reported in Section 6.3 are averaged across 10 perspective evaluations and do not directly measure

whether TRIAGE's top-$M$ candidates and generated patches are useful to a human practitioner. We complement them with a direct rating study on TRIAGE's top-3 output.

**Pre-registration.** Sample selection, the rating questions, and the analysis plan were fixed before any rating took place. **(1) Sample**: all 326 traces in the full validation pool, stratified across Who&When AG ($n$=123), Who&When HC ($n$=55), TRAIL GAIA ($n$=117), and TRAIL SWE-Bench ($n$=31), comprising every trace for which TRIAGE produced at least one top-$M$ candidate. 326 traces $\times$ up to 3 candidates $\approx$ 977 candidate-question rating units per rater. **(2) Questions**: For each candidate, Q1 step plausibility "Is this step a reasonable cause of the failure (would a practitioner reasonably investigate it)?" and Q2 patch plausibility "Is the generated patch a reasonable intervention?". Both on a 4-point scale: good / ok / poor / unsure. **(3) Reporting**: We report per-rater rating distributions, per-trace headline rates (top-1 good and top-3 any-good), per-stratum patch-good-or-ok rates, and inter-rater Cohen's $\kappa$. All ratings are reported, including negative findings.

**Raters.** One human rater and a panel of three frontier LLMs (Claude Opus 4.6, GPT-5.4, Grok 4.20; identical judges to the TriageBench council) rate all 977 candidate-question units on Q1 and Q2 under the same prompt and gold-context. The human pass took roughly 35 hours of focused work spread over multiple sessions; the LLM panel is a deliberate complement, not a substitute. LLM raters apply the rubric uniformly across all 977 units without attention drift, produce reproducible verdicts under temperature 0, and let us run three independent judges at marginal cost. The LLM panel rated each candidate independently with no cross-rater coordination. Verdicts are categorical (good / ok / poor / unsure), not pairwise; each candidate is rated on its own merits.

| Rater | Q | good | ok | poor | unsure | good+ok |
|---|---|---|---|---|---|---|
| Human | step | 38.1 | 51.5 | 9.1 | 1.3 | 89.6 |
| Human | patch | 24.4 | 54.6 | 19.7 | 1.4 | 78.9 |
| Opus 4.6 | step | 43.4 | 40.5 | 16.1 | 0.0 | 83.9 |
| Opus 4.6 | patch | 33.1 | 40.8 | 25.9 | 0.2 | 73.9 |
| GPT-5.4 | step | 45.9 | 40.8 | 13.3 | 0.0 | 86.7 |
| GPT-5.4 | patch | 36.5 | 35.6 | 27.4 | 0.4 | 72.2 |
| Grok 4.20 | step | 58.8 | 35.6 | 5.6 | 0.0 | 94.4 |
| Grok 4.20 | patch | 73.0 | 19.5 | 7.5 | 0.0 | 92.5 |

Table 25: Per-rater rating distributions (%) over 977 candidate units (326 traces, up to 3 candidates each).

**Results.** Table 25 reports rating distributions per rater per question across the 977 rating units. On patch plausibility, the human rates 24.4% *good* and 78.9% *good+ok*; the LLM panel ranges from 72.2% (GPT-5.4) to 92.5% (Grok 4.20) at *good+ok*, with Grok rating substantially more leniently than the other judges. Step plausibility ratings are higher than patch plausibility for every rater (human 89.6% vs. 78.9% *good+ok*; LLM panel 83.9 to 94.4% vs. 72.2 to 92.5%), consistent with patches being harder to judge than the underlying step identification.

| Rater | Q | $T1_g$ | $T1_{g+o}$ | $A3_g$ | $A3_{g+o}$ |
|---|---|---|---|---|---|
| Human | step | 60.1 | 92.6 | 84.7 | 98.8 |
| Human | patch | 34.0 | 86.5 | 56.7 | 98.5 |
| Opus | patch | 44.3 | 85.1 | 69.7 | 96.3 |
| GPT | patch | 45.7 | 76.7 | 78.5 | 97.2 |
| Grok | patch | 82.5 | 95.7 | 99.1 | 100.0 |
| Panel maj. (3) | patch | 52.1 | 86.2 | 83.7 | 99.1 |

Table 26: Per-trace top-1 and top-3-any patch plausibility rates (%) under strict (good only, subscript $g$) and lenient (good+ok, subscript $g{+}o$) thresholds. T1: top-1 patch matches threshold. A3: at least one of top-3 patches matches threshold. *Panel maj. (3)*: at least 2 of 3 LLM judges (Opus, GPT, Grok) rate the patch above threshold; this is the "panel" number cited in the abstract and §6.4.

**Per-trace headline rates.** Table 26 reports per-trace top-1 and top-3-any rates under both strict (*good* only) and lenient (*good+ok*) thresholds. The strict human rates are the most conservative headline (34.0% top-1, 56.7% top-3-any); the LLM-panel-majority lenient top-1 is 86.2% (at least 2 of 3 judges rating *good+ok*).

| Stratum | $n$ | $T1_g$ | $T1_{g+o}$ | $A3_g$ | $A3_{g+o}$ |
|---|---|---|---|---|---|
| W&W AG | 123 | 36.6 | 87.8 | 62.6 | 97.6 |
| W&W HC | 55 | 27.3 | 90.9 | 52.7 | 100.0 |
| TRAIL GAIA | 117 | 43.6 | 86.3 | 64.1 | 99.1 |
| TRAIL SWE-Bench | 31 | 0.0 | 74.2 | 12.9 | 96.8 |

Table 27: Per-stratum human patch plausibility (%). Strict (good only, subscript $g$) and lenient (good+ok, subscript $g{+}o$); top-1 and top-3-any.

**Per-stratum patch plausibility.** Table 27 reports per-stratum human rates. SWE-Bench is the weakest stratum for every conservative rater, consistent with the body's observation (Section 6.1) that long multi-error code traces stress the suspicion stage.

**Inter-rater agreement.** Table 28 reports Cohen's $\kappa$ between the human rater and each LLM judge. Opus and GPT show substantial agreement with

| | 4-class $\kappa$ | | Binary $\kappa$ | |
|---|---|---|---|---|
| Rater pair | step | patch | step | patch |
| Human vs. Opus | 0.709 | 0.677 | 0.630 | 0.706 |
| Human vs. GPT | 0.706 | 0.619 | 0.717 | 0.703 |
| Human vs. Grok | 0.403 | 0.137 | 0.320 | 0.351 |

Table 28: Cohen's $\kappa$ between the human rater and each LLM judge on the 4-class scale (good/ok/poor/unsure) and the binary collapse (*good+ok* vs. *poor+unsure*).

the human rater; Grok's much more lenient distribution (73.0% *good* on patches vs. the human's 24.4%) drives both its low $\kappa$ and the panel's higher top-1 lenient rates rather than indicating stronger agreement.

**Sample-size caveats.** The 326-trace pool yields 977 candidate-question ratings per rater, giving Wald 95% CIs of approximately $\pm$3pp on per-question rates and $\pm$5 to 9pp on per-stratum rates ($n = 31$ to 123). Only one human rater participates; the LLM panel partially mitigates this but shares the systematic LLM biases discussed in Section 7 (later-step preference, plausibility leniency relative to humans). Patch ratings, given access to the gold answer, may overestimate plausibility in a deployment setting where the practitioner does not have it; this is a known limitation of LLM-judge validation studies and motivates execution-grounded replay as a priority for future work.

# H Reproducibility

| Stage / item | Hyperparameter | Value |
|---|---|---|
| Headline model | provider / model id | moonshot / `kimi-k2.5` |
| | max output tokens | 8,192 |
| | concurrency / RPM / TPM | 100 / 5K / 3M |
| Alt. pipeline models | Sonnet 4 (Anthropic) | `claude-sonnet-4-20250514` |
| | Llama 3.3 70B (Together) | `meta-llama/Llama-3.3-70B-Instruct-Turbo` |
| TriageBench council | council member 1 | `gpt-5.4-2026-03-05` (OpenAI) |
| | council member 2 | `claude-opus-4-6` (Anthropic) |
| | council member 3 | `grok-4.20-0309-reasoning` (xAI) |
| Suspicion | passes $P$ | 5 |
| | aggregation | mean |
| | floor $\rho_{\min}$ | 0.01 |
| | temperature | provider default |
| Gating | top-$K$ floor / quantile | 8 / 0.25 |
| | replay budget $M$ | 3 |
| | anchor cap $\lvert A(i)\rvert$ | 6 |
| | mix coef. $\alpha$ | 0.7 |
| | criticality $\tau$ | 0.70 |
| Patch gen. | temperature | 0.0 (enforced) |
| | max retries | 2 |
| | max content length | 200K chars |
| | max diff ratio | 0.75 (advisory) |
| | default Lev. cap | 0.60 |
| Replay | temperature | 0.3 |
| | baseline / patched trials | 1 / 1 |
| | strategy | v3_perspective |
| | perspective id | 8 (locked) |
| | UCB $\beta, z, n_{\min}$ | 2.0, 1.96, 5 |
| Seeds | per-call seed | SHA256(*key*)[:8] |
| | metric reporting seed | 42 |
| Compute | API calls / tokens | 3,306 / $\sim$22.7M |
| | cost | $\sim$\$14.2 ($\sim$\$0.080/trace) |

Table 29: Non-default hyperparameters and run statistics for the headline Who&When run. Alt. pipeline models share these settings. Per-operator Lev. caps in Table 11.