TRIAGE: Severity-Ranked Multi-Agent Failure Attribution

Xizhi Wang
Troy High School

Abstract

Failure attribution for multi-agent LLM systems is typically formulated as identifying a single decisive error in a failed execution trace. Yet failed traces often contain multiple interacting errors and admit several plausible intervention points. To examine whether an annotated decisive error is also the most promising point for repair, we introduce TriageBench, comprising LLM-council judgments over 1,184 Who&When intervention candidates. The council prioritizes a different step from the original label in 59.3% of traces. We therefore reframe failure attribution as *severity-ranked diagnostic triage* and instantiate it as TRIAGE, a zero-shot, cost-aware cascade combining LLM-as-judge suspicion scoring, structural impact, and LLM-simulated replay over generated repair patches. Under top-K evaluation, TRIAGE 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). In a separate evaluation of 326 traces, strict ratings with gold-answer access identify at least one plausible top-3 repair on 56.7% (human rater) and 81.6% (three-LLM panel majority) of traces. Our code[1] and TriageBench[2] are publicly available.

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., 2025). 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 *earliest decisive error step*, the action whose correction would, counterfactually, have led the rest of the trajectory to the correct final answer (Zhang et al., 2025). 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 at the time of the study, leading attribution methods reported overall step-level accuracy of around 40% on Who&When (Tables 1 and 2).

Who&When acknowledges that a trace may contain multiple decisive errors, but selects the earliest as its attribution target (Zhang et al., 2025). This choice leaves the usefulness and relative priority of alternative intervention points unevaluated. On cases flagged as uncertain, its annotators initially disagreed at rates of 9.1 to 50% (Zhang et al., 2025). We introduce TriageBench to study alternative repair priorities through judgments from a frontier-LLM council. The council prioritizes a different step from the original annotators in 59.3% of traces under a repair-priority criterion (Section 7).

A Who&When logic puzzle illustrates why a failed trace can offer several useful intervention points.

> **Task (summarized).** *Among 100 residents, humans always tell the truth and vampires always lie; at least one resident is known to be a vampire, yet everyone says, "At least one of us is a human."*

A vampire's statement must be false, which implies

[1] https://github.com/jimmywang585/triage-failure-attribution

[2] https://huggingface.co/datasets/jimmywang585/triage-bench

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 376–397
November 6-10, 2026 ©2026 Association for Computational Linguistics

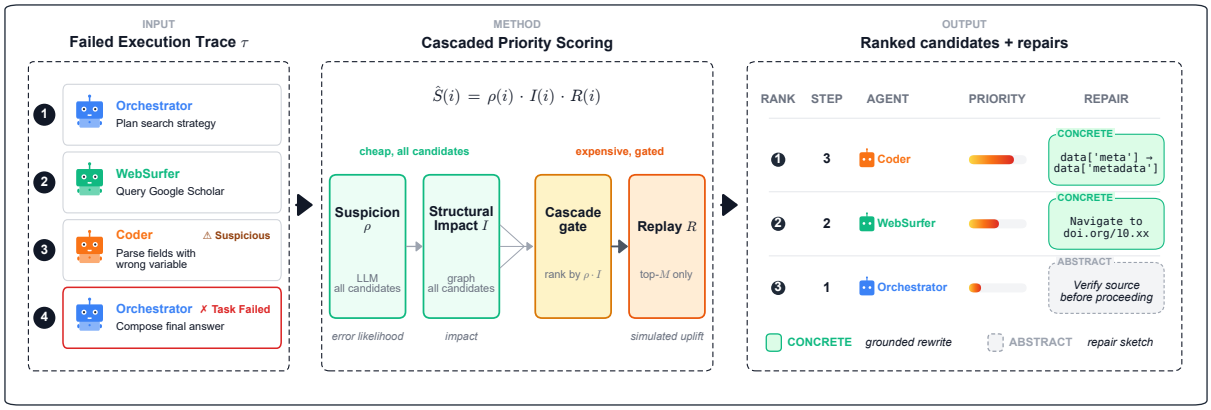


Figure 1: TRIAGE pipeline. Suspicion (ρ) and structural impact (I) score all candidates; repair generation and LLM-simulated replay are gated to the top-M. The heuristic priority score $\hat{S} = \rho I R$ uses the replay multiplier R, with $R = 1$ for other candidates. Output: ranked intervention candidates with repair proposals for the top-M.

that there are no humans; hence all 100 residents are vampires. In the failed trace, a reasoning agent incorrectly rejects the all-vampire solution because the residents' statements would be false, overlooking that vampires must lie. It concludes that there are no vampires, and a validation agent endorses the same error. Who&When labels the reasoning step. TRIAGE ranks that step first and the validation step second, generating a concrete correction at each: one derives the all-vampire solution, while the other corrects the validator's mistaken rejection of it. If a method selected the validation step as its sole prediction, Who&When would mark that prediction incorrect, even though the generated patch corrects a substantive logical error there. A valid single-step label can still leave useful correction points uncredited. 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.

We define *severity* as a step's intervention value (Section 3) and propose TRIAGE, a zero-shot, cost-aware diagnostic cascade that uses a heuristic priority score to rank intervention candidates. The cascade combines three per-step signals: an LLM-judged suspicion score $\rho \in [0, 1]$ for each step, 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. Under strict (*good*-only) rating, one human rater with access to the gold answer finds at least one plausible top-3 repair on 56.7% of 326 traces (Section 6.4), pairing localization with inspectable repair hypotheses.

Ranked output allows practitioners to inspect several candidate interventions rather than rely on a single prediction. On Who&When, TRIAGE includes the annotated error among its top-5 candidates on 73.9% of traces, compared with 42.4% at top-1. On multi-error TRAIL, coverage reaches 93.2% at top-10, compared with 56.8% at top-1 (Section 6.2).

Our contributions are: (1) we reformulate multi-agent failure attribution as severity-ranked diagnostic triage, motivated by useful correction points beyond a single decisive-step label, and show that top-K evaluation reveals substantially more diagnostic 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 judgments and consensus rankings over 1,184 candidate intervention targets on Who&When, alongside the original decisive-step labels.

2 Related Work

2.1 MAS Failure Modes and Taxonomies

Cemri et al. (2025) analyzed 1,600+ annotated traces across seven MAS frameworks and introduced MAST, a taxonomy of 14 failure modes spanning specification/system design issues, inter-agent misalignment, and task verification. Other studies analyze error propagation (Ma et al., 2026b; Barrak, 2025) and report task-recovery gains from root-cause correction (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., 2025). The original Who&When baselines (all-at-once, step-by-step, binary search) are prompting strategies; reproduced on the stronger Kimi K2.5 model, step-by-step rises to 26.6% (Table 1), still well below TRIAGE's 42.4% on the same model (Section 6.1).

One line structures traces for attribution: ECHO decomposes traces into hierarchical context levels (Banerjee et al., 2025), and CHIEF combines hierarchical causal graphs, backtracking guided by virtual oracles, and counterfactual screening (Wang et al., 2026). CDC-MAS combines Shapley-based agent attribution with causal discovery and counterfactual improvement estimates for step ranking (Ma et al., 2025). Another line narrows candidates: RAFFLES employs aspect-specific evaluators over successive judging rounds (Zhu et al., 2026), while SDBL delineates a failure-containing scope before reasoning within it (Sun et al., 2026). 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 and ordinal impact levels, reporting an average of 5.7 errors per trace across 148 executions. TRIAGE explores a complementary reframing, taking intervention value as the objective for ranking repair candidates.

MP-Bench (In et al., 2026) preserves multiple human attributions and evaluates rankings against annotation frequency. TraceElephant (Chen et al., 2026) provides full execution context and reproducible environments for failure attribution. TriageBench complements these resources with pairwise judgments of repair priority: which local correction would most plausibly improve recovery, rather than how frequently a step is marked erroneous.

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 evaluates multiple viable repairs through checkpointed re-execution, measuring task recovery and progress (Ma et al., 2026a). ErrorProbe (Li et al., 2026) combines backward tracing with tool-grounded hypothesis verification and retains verified diagnoses in episodic memory. Other approaches simulate counterfactual reasoning in a single LLM call (West et al., 2025) or use counterfactual replay to construct attribution data for reinforcement learning (Zhang et al., 2026). TRIAGE generates a repair patch for each selected candidate and scores the original and patched trace contexts in separate LLM calls, 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. (2025). 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 τ', 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 τ by *severity*, defined as the intervention value of a step.

Definition 1 (Severity)**.** *For step i in a failed trajectory τ, 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 τ' 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 diagnostic

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

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

sorted by decreasing $S(i)$, where r_i is a candidate repair for step i. The top-ranked candidate need not be the earliest decisive error 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 uses the cascaded priority score introduced in Section 4 to rank candidate intervention targets.

4 Method

TRIAGE combines three signals relevant to intervention priority: whether a step is likely erroneous, how its error may affect downstream execution, and whether a proposed repair could improve the outcome. Suspicion and structural impact prioritize candidates for repair generation; simulated replay then provides repair-specific evidence for the selected candidates. We combine these signals into a continuous priority score:

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

where $\rho(i) \in [0, 1]$ scores step i as a possible earliest root-cause error, $I(i) \in [0, 1]$ estimates the structural propagation of that error, and $R(i) \geq 1$ is a replay-based promotion multiplier. We treat $\hat{S}$ as a pragmatic repair-priority ranker, not a calibrated estimator of the causal severity $S(i)$ from Definition 1.

The pipeline computes $\hat{S}$ in four stages: trace normalization, suspicion scoring (ρ), structural impact (I), and repair generation with LLM-simulated replay (R) over a top-M subset. Candidates are ranked by $\hat{S}$. For single-label evaluation, the top-ranked candidate is the decisive-error prediction; for diagnosis, we return a ranked list of steps to inspect, together with repair proposals for the top-M candidates.

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 (ρ)

The suspicion stage uses LLM-as-judge scoring (Zheng et al., 2023; Li et al., 2024): the LLM receives the normalized trace and assigns a suspicion score $\rho^{(p)}(i) \in [0, 1]$ to every step in a single pass. To reduce sampling variability, the default configuration averages P=5 independent scoring passes for each step:

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

4.2 Structural Impact (I)

Trace-impact graph. An error can remain locally contained or propagate into later decisions. To approximate these consequences for each candidate i without additional LLM calls, we use a heuristic trace-impact graph, drawing on program-slicing intuitions (Weiser, 1984) for LLM-agent traces. 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 $I_{\text{graph}}(i) \in [0, 1]$ approximates the potential downstream propagation of an error at i. Full equations, the step-type weight table, and the anchor-selection rules are in Appendix B.

Position-prior mix. Traces with few downstream anchors provide limited structural information (e.g., short retrieval traces). To stabilize ranking in this regime, we mix I_{graph} with a position-decay prior π:

$$\begin{aligned} I(i) &= \alpha I_{\text{graph}}(i) + (1-\alpha)\,\pi(i), \\ \pi(i) &= \tfrac{1}{1+i}. \end{aligned} \tag{6}$$

Intuitively, earlier corrections can influence more subsequent actions; the harmonic $\frac{1}{1+i}$ encodes this as a smooth, monotone heuristic position prior.

4.3 Repair Generation and Replay (R)

Repair generation turns a candidate location into a repair hypothesis, which replay assesses for its potential effect on the outcome. 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 an LLM evaluator.

Candidate selection. We restrict repair generation and replay 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).

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., 2025), and a *patch* call applies a constrained operator paired with the inferred category to generate the repair. 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").

LLM-simulated replay. "Replay" here is LLM-simulated scoring of a patched trace, not re-execution of the multi-agent system. For each $i \in \mathcal{C}_M$, an LLM evaluator estimates the probability of eventual task success 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 original and patched contexts are scored in separate calls. Abstract repairs are identified as hypothetical corrective sketches. 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 5 lists ten interchangeable choices). Execution-grounded validation (re-running agents under the patched step) is left for future work (see Limitations). The replay multiplier is:

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

Replay is promotion-only: non-positive simulated uplift is ambiguous because the patch may be incomplete or the evaluator insensitive even when the step is causal, so we floor R at 1. Appendix C.4 discusses this design choice.

Replay budget. 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.

5 Experiments

5.1 Datasets

Who&When. The Who&When benchmark (Zhang et al., 2025) 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 (52.5 vs. 8.7 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 evaluate whether the top-K candidates include an annotated error location; we do not report TRAIL's error-category F1 or joint accuracy (category and location).

5.2 Metrics

We report *step accuracy* (top-1, following Zhang et al., 2025; Zhu et al., 2026), *top-K recall* for ranked output, and *NDCG@3* for the component ablation; MRR and within-2 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.

	Llama 3.3 70B			Kimi K2.5			Claude Sonnet 4		
Method	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	31.7	15.5	26.6	9.5	0.0	6.5
Binary Search	14.3	3.4	10.9	24.6	17.2	22.3	19.8	**12.1**	17.4
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.

5.3 Baselines

Prior methods. On Who&When, we compare against the main reported 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), with the corresponding models and configurations listed in Table 2. We compute All as the 126:58 weighted mean of the reported AG/HC accuracies. SDBL uses 54 HC traces; CDC-MAS uses an 80/10/10 train/validation/test split.

Same-LLM baselines. To isolate method from model, we re-run Who&When's three published prompting strategies (Step-by-Step, Binary Search, All-at-Once) (Zhang et al., 2025) on each LLM. We also evaluate TRIAGE's suspicion stage in isolation (ρ-only; Appendix E.3). For ranked output, *LLM-judge (LJ)* is a single-call baseline that directly returns a ranked list of up to ten candidate error steps (Appendix A.5). We use the same LJ prompt and model (Kimi K2.5) on both datasets, scoring predictions against the single annotated step on Who&When and any annotated error location on TRAIL. Unlike the suspicion stage, LJ directly generates the list without scoring every step.

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). 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). Per-trace cost: Appendix F.

Hyperparameters. We set α=0.7, suspicion floor 0.01, and anchor cap $|A(i)|\leq 6$ by inspection on Who&When. The top-K and formula analyses use retained candidate pools and pre-replay ρI rankings, with one suspicion pass on Who&When and P=5 on TRAIL. Exact model IDs are listed in Table 23; Levenshtein caps, graph weights, and formula ablation are in Appendices C, B, E.5.

6 Results

This section reports top-1 accuracy across models (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).

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: Main reported step accuracy (%) of prior methods on Who&When. RAFFLES: K=2; ECHO: full I4; SDBL: EASD. Bold/underline: best/second-best per column.

6.1 Top-1 Across Models

TRIAGE improves top-1 localization over the original prompting strategies. In our same-LLM reproductions, TRIAGE exceeds all three prompting baselines on the overall Who&When split for each model, reaching 29.3% with Llama 3.3 70B, 42.4% with Kimi K2.5, and 37.5% with Claude Sonnet 4 (Table 1). The gains are largest on AG, where TRIAGE leads the prompting baselines across all three backbones. HC remains more difficult: TRIAGE exceeds the prompting baselines on Kimi K2.5 and matches Binary Search on Sonnet 4. Thus, the overall gains are consistent across backbones, but the benefit varies across trace types.

TRIAGE is competitive with leading attribution methods. Table 2 summarizes the main reported step accuracy of prior methods. At 42.4%,

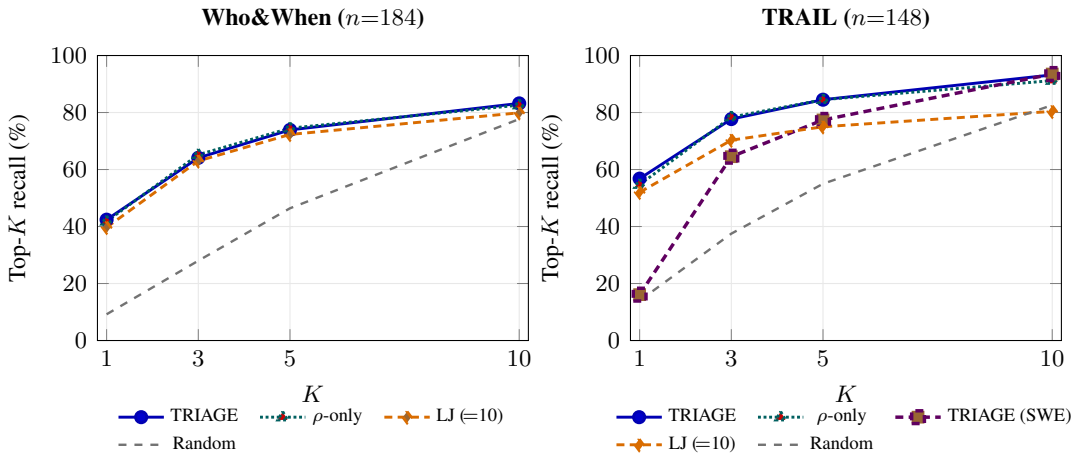


Figure 2: Top-K step recall on Who&When (n=184) and TRAIL (n=148, any-error hit) using Kimi K2.5. ρ-only: TRIAGE's suspicion stage alone. LJ (=10): single-call top-10 ranked list. TRIAGE: pre-replay ρI ranking. TRIAGE (SWE): SWE-Bench subset of TRAIL (31 traces).

TRIAGE with Kimi K2.5 matches the highest overall accuracy in Table 2, achieved by RAFFLES with Claude Sonnet 4.

6.2 Top-K Coverage

Top-K recovers more annotated error targets than top-1. 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, accuracy rises to 93.2% top-10 vs. 56.8% top-1, a +36.4pp gap. The top-ranked step can miss an annotated error that still appears among the next few candidates. At the same inspection budget of K=5, TRIAGE exceeds uniform random selection on both Who&When (73.9% vs. 46.4%) and TRAIL (84.5% vs. 55.0%; Appendix Table 12).

The TRAIL subsets show different gains from inspecting multiple candidates. GAIA reaches 67.5% top-1 and 86.3% top-5; SWE-Bench is harder at top-1 (16.1%) but reaches 77.4% top-5. The larger gain on SWE-Bench shows that a short candidate list can retain substantial localization signal even when the first choice is unreliable. The comparison does not isolate context length from differences in task domain and error structure.

Comparison with directly prompted rankings. On TRAIL, TRIAGE reaches 93.2% top-10 against LJ's 80.4% (+12.8pp); the paired-bootstrap comparison is statistically significant (Appendix E.4). On Who&When, TRIAGE and LJ reach 73.9% and 72.3% at top-5; their paired differences are not statistically significant at any reported K. Section 6.3 examines which cascade components account for these results.

6.3 Cascade Contribution

The cascade turns ranked candidates into inspectable repair hypotheses. Beyond ranking, TRIAGE generates repair patches and simulated-uplift estimates for its selected candidates (Section 6.4). Its nominal per-trace budget (≈17 LLM calls: 5 suspicion + 6 patch + 6 replay; Appendix F) is higher than lightweight ranking baselines. These additional calls produce inspectable repair proposals and estimates of their effect.

	W&W		TRAIL	
	T1	T10	T1	T10
ρ-only	41.8	82.6	54.7	91.2
ρI	**42.4**	**83.2**	**56.8**	**93.2**

Table 3: Suspicion and impact ranking (Kimi K2.5) on Who&When (n=184) and TRAIL (n=148), top-1 (T1) and top-10 (T10) step accuracy (%).

Per-step scoring carries most of the localization signal. The suspicion-only and ρI rankings are close at top-1 on both benchmarks (Table 3). On TRAIL, suspicion alone reaches 91.2% top-10 against LJ's 80.4%, accounting for most of TRIAGE's gain (Appendix E.3).

Structural impact (I) gates candidates without additional LLM calls. Adding I to ρ improves Who&When top-1 by +0.6pp and TRAIL top-1 and top-10 by +2.1pp and +2.0pp, respectively (Table 3). These gains are small, and the

cascade trades off slightly at intermediate K (Appendix E.3). Beyond ranking, I gates which candidates receive patches by combining suspicion with downstream propagation structure.

Structural modeling and replay address aspects of intervention value that error likelihood alone does not capture: the downstream consequences of a mistake and the effect of a proposed correction. The present heuristic graph and simulated evaluator are initial approximations; richer dependency models and execution-grounded replay remain to be evaluated against the severity objective.

6.4 Patch Plausibility

Under strict rating, the top-3 patch set contains a good-rated repair on a majority of traces. The patch-rating study evaluates separately generated diagnostic outputs on 326 traces (178 Who&When, 148 TRAIL). A human rater and a three-LLM panel (Verga et al., 2024) comprising GPT-5.4, Claude Opus 4.6, and Grok 4.20 rated patch plausibility on a 4-point scale, with access to the gold answer and rating each candidate independently (Appendix G). Top-3 any-plausible is 57% / 82% (human / 3-LLM panel majority) under strict (*good*-only) and 98.5% / 99% under lenient (*good* + *ok*); top-1 strict is 34% / 52% and top-1 lenient 86.5% / 86.2% (Table 20). Human–LLM agreement on binary patch plausibility ranges from $\kappa = 0.35$ to 0.71, with lower agreement for Grok than for Opus and GPT. The 23pp gap between strict top-1 (34% human) and strict top-3 (57% human) supports inspecting multiple candidates: a top-3 set can contain a good-rated repair when the first candidate does not. TRIAGE thus pairs ranked diagnosis with an inspectable repair artifact: each trace yields up to three intervention candidates, each paired with a generated repair and a replay-based uplift estimate.

7 TriageBench: Judging Intervention Priority

Evaluating severity requires comparing the intervention value of candidate steps, beyond checking whether a ranking contains an annotated error. We introduce TriageBench as an initial resource for this comparison: council judgments of repair priority alongside Who&When's original decisive-error labels. The study examines whether the two criteria favor the same first choice and how TRIAGE's rankings align with the council's priorities.

Candidate interventions. TriageBench covers 177 Who&When traces from both AG and HC, with 1,184 candidate steps (mean 6.7 per trace). Each pool contains up to eight candidates and always retains the original human-annotated decisive-error step. Judges assess the potential of locally correcting each candidate. The assumed correction uses only information available up to that step, keeps the preceding trace fixed, and allows downstream execution to resume normally.

Council judgments. We use a multi-model council (Zhao et al., 2025) comprising GPT-5.4, Claude Opus 4.6, and Grok 4.20 (exact model IDs in Table 23). Each judge receives trace context, candidate steps, and the gold final answer. For every pair of candidate steps within a trace, each judge compares which of the two local corrections would most plausibly increase the probability of recovering that answer. The prompt distinguishes this criterion from earliest-error identification and prohibits favoring answer-adjacent steps merely because the known answer could be inserted there. Judges separately assess candidate plausibility and recovery potential, then provide pairwise preferences, ties or abstentions, and rationales. To reduce position bias, each judge evaluates the pair twice, swapping which candidate is presented first. A Davidson tie-aware Bradley–Terry (BT) model (Davidson, 1970) combines these preferences into a ranking of the candidates within each trace, with relative repair-priority scores and posterior uncertainty. Gold-answer access supports the annotation task; TRIAGE's attribution pipeline does not receive it.

Repair priorities and decisive-error labels. All three judges agree on the preferred candidate or on a tie in 74.1% of comparisons with three recorded judgments (1,228/1,658); pairwise winner agreement reaches Fleiss $\kappa = 0.646$. Despite this agreement, the consensus first choice differs from the original decisive-error label on 59.3% of traces. Because that label is retained in every pool, these differences are not caused by its exclusion. The original step remains within the council's top-3 on 68.9% of traces and top-5 on 84.2%. The rankings thus express alternative priorities over overlapping targets. Since both the criterion and annotators differ, disagreement alone does not establish that the council is correct or identify an accuracy ceiling for Who&When. Appendix D.4 reports agreement and posterior diagnostics.

Comparison with TRIAGE. Using the Kimi predictions reported in Table 1, TRIAGE's first choice matches the council's on 32.8% of TriageBench traces (58/177). Agreement reflects both candidate coverage and prioritization: among the 136 traces whose TRIAGE prediction is in the council's pool, agreement is 42.6% (58/136). Over shared candidates, TRIAGE's pre-replay suspicion–impact scores from the same run have a mean within-trace Spearman correlation of 0.271 with council scores across 168 eligible traces. This modest alignment shows that TRIAGE and the council differ in both their preferred steps and their ordering. The comparison examines intervention priorities that localization accuracy alone leaves unmeasured, while neither ranking has been validated against recovery under executed interventions. Appendix D.5 details the comparison.

A resource for severity-ranked attribution. TRIAGE supplies a practical ranking method and repair hypotheses; TriageBench supplies comparative judgments with which to examine candidate priorities. The release includes the original traces and labels, candidate rankings, posterior uncertainty, individual judge verdicts and rationales, and detailed pairwise comparisons. These records support comparing rankers, inspecting why particular corrections receive priority, and studying disagreements or alternative aggregation methods. The scores represent judged repair priority, rather than direct measurements of $S(i)$ in Definition 1. A direct next test is whether higher-ranked candidates yield greater task recovery under a common repair procedure and execution budget. Approaching the severity objective additionally requires searching over repairs at each step and estimating the best repairs' recovery probabilities through repeated executions. Appendix D gives the construction protocol and prompt.

8 Conclusion

Single-label failure attribution captures only part of the diagnostic task: a failed trace can offer several useful correction points. Severity makes their relative intervention value an explicit objective, defined by the best-case probability of averting failure through a local repair. Our results show that per-step suspicion scoring provides a strong basis for ranked diagnosis, remaining close to the impact-weighted ranking in localization accuracy on both benchmarks. Top-K evaluation reveals substantially more annotated error targets than top-1. TRIAGE pairs its top-M=3 candidates with generated repair patches and simulated-replay uplift estimates. In the separate patch study, top-3 repair sets contain a plausible repair on 56.7% of 326 traces under strict (*good*-only) rating by one human rater. TriageBench supplies reusable repair-priority judgments and shows that the council often favors a different correction point from the original decisive-error label.

Reliable severity estimation remains an open problem. TRIAGE relies on heuristic impact and simulated replay, while TriageBench relies on council judgments; neither has been validated against recovery under executed interventions. The next step is to test whether higher-ranked candidates yield greater task recovery under a common repair procedure and execution budget. Better repair search and repeated execution are needed to estimate intervention value more faithfully, with evaluation across a broader range of systems and failure types. We advocate adopting severity as a central objective for failure attribution: methods should be evaluated on how well they prioritize interventions that avert failure, alongside agreement with existing decisive-error labels.

Limitations

Validation of intervention value. Our evaluation measures localization, patch plausibility, and agreement with council judgments, but does not establish whether higher-ranked interventions improve task recovery. TRIAGE uses LLM-simulated replay, and neither its ranking nor TriageBench's preferences have been validated through executed repairs. The patch study includes one human rater and an LLM panel with access to the gold answer, limiting conclusions about broader human agreement and repair usefulness in deployment.

Scope of the evidence. TriageBench's judgments are limited to Who&When traces and their selected candidate pools. Differences from the original labels reflect changes in both the attribution criterion and the annotators; they do not establish that the council is correct or quantify an accuracy ceiling. Some comparisons with published methods use different backbone models, so performance differences cannot be attributed solely to the attribution method.

Ethical Considerations

This work uses existing multi-agent benchmark traces. The author performed all human ratings; no external participants were recruited. TRIAGE's rankings, generated repairs, and TriageBench judgments may be incorrect or reflect model biases. They should support human review, and proposed repairs require validation through execution before deployment. Applying the method to private traces requires safeguards for sensitive information shared with model providers. Reuse and redistribution of benchmark material should respect the original licenses and access conditions.

The author used AI tools to assist with language polishing, LaTeX formatting, and paper organization. The author takes full responsibility for the content and results of this paper.

Acknowledgments

I am very grateful to Professor Xiaowu Dai of UCLA for his mentorship throughout this paper. The time he devoted to our regular discussions helped me to develop and refine this work greatly.

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 A Zaharia, Joseph Gonzalez, and Ion Stoica. 2025. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc.

Mengzhuo Chen, Junjie Wang, Fangwen Mu, Yawen Wang, Zhe Liu, Huanxiang Feng, and Qing Wang. 2026. Seeing the whole elephant: A benchmark for failure attribution in LLM-based multi-agent systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19888–19905, San Diego, California, United States. Association for Computational Linguistics.

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 *International Conference on Learning Representations*, volume 2024, pages 23247–23275.

Yeonjun In, Mehrab Tanjim, Jayakumar Subramanian, Sungchul Kim, Uttaran Bhattacharya, Wonjoong Kim, Sangwu Park, Somdeb Sarkhel, and Chanyoung Park. 2026. Rethinking failure attribution in multi-agent systems: A multi-perspective benchmark and evaluation. *Preprint*, arXiv:2603.25001.

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 *International Conference on Learning Representations*, volume 2024, pages 54107–54157.

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. In *International Conference on Learning Representations*, volume 2026, pages 52994–53028.

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.

Jiazheng Li, Emine Yilmaz, Bei Chen, and Thu Le. 2026. Towards self-improving error diagnosis in multi-agent systems. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 2063–2077, San Diego, California, United States. Association for Computational Linguistics.

Guoqing Ma, Jia Zhu, 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. *arXiv 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. In *International Conference on Learning Representations*, volume 2026, pages 28059–28087.

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 *International Conference on Learning Representations*, volume 2024, pages 9025–9049.

Judea Pearl. 2009. *Causality*. Cambridge University Press.

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.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing judges with juries: Evaluating LLM generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*.

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 511–518.

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.

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

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.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. 2026. AgenTracer: Who is inducing failure in the LLM agentic systems? In *International Conference on Learning Representations*, volume 2026, pages 11377–11399.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. 2025. 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*.

Justin Zhao, Flor Miriam Plaza-del Arco, Benjamin Genchel, and Amanda Cercas Curry. 2025. Language model council: Democratically benchmarking foundation models on highly subjective tasks. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12395–12450, Albuquerque, New Mexico. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

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.

A Prompts and Implementation Details

Pipeline stages use a common normalization of the task and trace. Suspicion scoring and patch generation receive the full normalized trace; replay receives the context through the candidate checkpoint (Appendix A.4).

The prompt templates 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 (Appendix D.3) receives the gold answer when judging repair priority. The prompt excerpts below omit repeated context and show conditional fields schematically; the complete TRIAGE templates are available in the released code.

A.1 Trace normalization

Raw traces (Who&When, TRAIL) are normalized into a uniform sequence of step records. Each record contains a step ID, agent name, structural role (`plan`, `tool_call`, `tool_result`, `factual_assertion`, `tool_interpretation`, `conclusion`, `delegation`, `other`), and full content.

Step roles are inferred by a rule-based classifier from the agent's message and used in the structural-impact graph. Table 4 reports post-normalization trace statistics.

Split	n	mean	median	p25	p95
W&W AG	123	8.7	10	7	10
W&W HC	55	52.5	33	20	129
W&W All	178	22.2	10	8	107.1
TRAIL GAIA	114	31.0	15.5	11	87.5
TRAIL SWE	30	34.2	32	24	54
TRAIL All	144	31.7	20	12.5	83.4

Table 4: Normalized trace length (steps) for 178 Who&When traces and 144 TRAIL traces with gold annotations. p25 and p95 use linear interpolation. Headline accuracy uses the full benchmark denominators (n=184 W&W, n=148 TRAIL).

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. Scores are clipped to $[0.01, 1]$ to retain a non-zero suspicion score for each candidate. The default configuration averages per-step scores over $P = 5$ independent passes.

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 debugging priors (`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., 2025) without these priors.

A.3 Patch generation prompts

Patch generation uses a *hypothesis* call to classify the candidate's error, followed by a *patch* call to emit the rewrite; validation failures can trigger retries (Appendix C). Both calls receive the same normalized task and trace context. The hypothesis call returns the inferred error category and a short rationale. Its prompt lists the operational categories and a category-selection rubric. The residual unknown class in Table 8 is used when no error category can be assigned.

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.
```

Hypothesis prompt (continued)

```
- 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.
```

Operator descriptions (patch call) (continued)

```
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.

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.
```

Patch prompt body (continued)

```
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

Replay uses one evaluator call for the original step and one for its patch, under a single perspective from Table 5. Each call presents context through the candidate checkpoint and asks the evaluator to estimate eventual task success. The baseline context retains the original candidate step; the patched context substitutes its repair, with all preceding steps unchanged. Abstract repairs are marked as hypothetical corrective sketches and include a short summary of the original step.

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 resulting uplift multiplier is $R(i) = 1 + \max(0, \hat{p} - p_0)$. Appendix C.4 explains the promotion-only design.

The ten perspectives are listed in Table 5. Each uses a different preamble and rubric with the same

task description, trace context, and step under evaluation. The pipeline does not supply the evaluator with the ground-truth answer.

id	Perspective	Rubric focus
0	Factual correctness	Is content supported by the preceding trace or grounded sources?
1	Downstream trajectory	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 the step propagate or correct upstream errors?
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	Information sufficiency	Is enough grounded information present to proceed?
7	Alternative assessment	Does it miss a clearly better, reasonable action?
8	Critical error detection	How severe and recoverable are the step's errors?
9	Holistic trajectory quality	Whole-trajectory composite of recovery likelihood.

Table 5: The ten replay perspectives. TRIAGE uses critical error detection.

A.5 LLM-as-judge baseline prompt

The single-call LLM-as-judge baseline (Figure 2, Table 13) receives the same normalized task and trace as suspicion scoring. It asks for an ordered list of candidate error steps, enabling top-K comparison with TRIAGE. The target list length is $\min(10, |\tau|)$, where $|\tau|$ is the number of normalized steps; the excerpt below shows the instruction for a target of 10.

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.
```

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

```
## 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, ...]
```

The baseline uses the same Kimi K2.5 model as TRIAGE's suspicion stage. Of 178 evaluable Who&When traces, 175 yielded usable rankings (the remaining 3 counted as misses against n=184). On TRAIL, 132 of 148 traces yielded usable 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 select previously unselected downstream anchors $A(i)$ in this order: the next non-tool response from another agent, next tool action, next tool interpretation, next plan, last termination step, and last non-termination conclusion or factual assertion. We then scan remaining tool-interpretation or conclusion steps chronologically, adding those at or above the trace-wide 70th-percentile suspicion threshold. Eligible anchors exclude human-authored steps and TRAIL container spans. We deduplicate by step ID and retain at most six anchors in selection order. If no listed type is found, we use the next eligible step; if no eligible downstream step exists, we use a self-anchor, $A(i) = \{i\}$ with $s_{ii} = 1$. For each non-self anchor $j \in A(i)$, edge sensitivity is

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

where the logarithmic distance decay is

$$d_{ij} = \frac{1}{1 + \log\big(1 + |j - i|\big)}.$$

The step-family pair weight is f_{ij} (Table 6), and $\beta = 0.3$ is a heuristic boost for steps produced by the same agent.

B.2 Criticality and aggregation

Each anchor j receives

$$c_j = \mathrm{clip}_{[0,1.5]}\Big(c_j^{\text{type}}\big(1 + \max\{0, \rho(j)\}\big)\Big)\,, \quad (9)$$

where c_j^{type} is a base weight by step family (Table 7). Termination anchors use a base weight of 1.0 regardless of inferred family. All anchors receive the suspicion multiplier and the 1.5 cap, giving greater weight to propagation through suspicious downstream steps.

The aggregated graph score is

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

mixed with the position prior as

$$I(i) = \alpha I_{\text{graph}}(i) + \frac{1-\alpha}{1+i}.$$

The position-mix coefficient is $\alpha = 0.7$. 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	–	0.9	–	0.6	0.5
Tool-interp.	0.5	–	–	0.7	0.6
Plan	0.8	0.5	–	0.6	0.5
Factual	0.5	–	–	0.7	0.5
Delegation	0.6	–	–	0.5	0.6
Other	–	–	–	0.5	0.5

Table 6: Step-family pair weights f_{ij} for selected anchor types. A dash denotes the default weight, 0.4, which also applies to unlisted pairs. T-act = tool action, T-int = tool interpretation.

Family	c^{type}	Family	c^{type}
Conclusion	0.9	Tool-action	0.6
Tool-interp.	0.7	Factual	0.6
Plan	0.7	Delegation	0.5
Other	0.5	Termination	1.0

Table 7: Base criticality weights c^{type}, before applying the suspicion multiplier and cap. Termination overrides the inferred family with a base weight of 1.0.

C Repair Generation

C.1 Error taxonomy and patch operators

TRIAGE's patch generation uses a hand-designed taxonomy covering failure modes observed in MAST-Data (Cemri et al., 2025) and Who&When traces, with a residual `unknown` category for steps the classifier cannot place. A patch operator specifies an edit type, such as correcting reasoning or fixing a tool call; valid operators are those allowed for the diagnosed error category. Each of the 13 categories maps to a *primary* patch operator and, where appropriate, additional valid operators (Table 8). The hypothesis call selects the error category; the patch call then selects an operator allowed for that category, or `OTHER`. This separates error diagnosis from repair generation and constrains the proposed edit to the diagnosed error type.

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 8: 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.

Table 9 shows the empirical class and operator distribution over the 530 patches generated for the headline Who&When run.

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 9: Hypothesis-class and operator distributions over 530 patches (178-trace Who&When run). Abstract repair is the rule-based fallback.

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"). We use abstract repairs when no grounded operator applies and identify them as sketches during evaluation.

C.3 Patch checks and fallback rules

Generated patches are checked against per-operator normalized edit-distance caps (Table 10). We divide the character-level Levenshtein distance by the longer of the original and patched text lengths. Patches above the cap are rejected and regenerated with feedback, with up to two retries 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), the required output format for structured steps, and signs of fabricated 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 10: Per-operator caps on normalized Levenshtein edit distance.

C.4 Promotion-only replay design

The replay multiplier

$$R(i) = 1 + \max\{0, \hat{p}(i) - p_0(i)\}$$

lies in $[1, 2]$ for replayed candidates and equals 1 otherwise. A non-positive simulated uplift leaves the candidate at its $\rho \cdot I$ baseline.

Rationale. A patched success estimate $\hat{p}$ may fail to exceed p_0 because the candidate is not a useful intervention target, the patch is incomplete, or the evaluator is insensitive to the change. A negative difference therefore cannot reliably distinguish a poor candidate from a poor patch or evaluation.

About 21% of repairs are abstract fallbacks (Table 9), which specify an intended change rather than a grounded rewrite. Even grounded patches may not receive positive simulated uplift. The floor retains these candidates for inspection when replay supplies no positive evidence.

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). Eligible steps are selected by content-bearing, causal-relevance, and executability filters. Short traces retain all eligible steps; larger pools are capped at eight candidates. The original decisive-error step is retained even if it fails these filters.

D.2 Pairwise annotation and Bradley-Terry aggregation

Each council member receives trace context and the ground-truth answer. Judges separately assess each candidate's plausibility (yes / no / uncertain) and answer-recovery potential (high / medium / low / none / uncertain), with a rationale. A pair consists of two candidate steps from the same trace: the judge compares the potential effect of locally correcting one versus the other. For each pair, the judge returns `A_strong`, `A`, `tie`, `B`, `B_strong`, or `insufficient_evidence`, with a brief rationale. Each pair is presented twice, swapping which candidate is labeled A and shown first. Votes are mapped back to the original candidate IDs; a judge's conflicting preferences across the two orders are excluded, while a single available order is retained. A tie expresses similar expected effect; `insufficient_evidence` is an abstention.

Pairwise votes are aggregated via the Davidson model (Davidson, 1970), a Bradley–Terry extension with an explicit tie parameter ν. We fit the model by maximum a posteriori (MAP) estimation with unit-scale Gaussian priors on the sum-to-zero candidate strengths and the log tie parameter. Consensus strengths are the MAP estimates; intervals and top-1 probabilities use 2,000 draws from a Laplace approximation to the posterior. Candidates rejected by the plausibility consensus are excluded from the fit and placed after the fitted candidates in the ranking; their BT scores are null. A top-1 posterior probability measures uncertainty about the leading candidate under this preference model, not the probability that its repair succeeds.

D.3 Council prompt

Each council member receives trace context, 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 distinguishes this from root-cause or earliest-error identification and prohibits positional preference without a downstream-recovery justification. The required `bias_check` field 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:
{
  "analysis": {
    "candidate_a_effect": "...",
    "candidate_b_effect": "...",
    "bias_check": "..."
  },
  "winner": "A_strong | A | tie | B | B_strong |
      insufficient_evidence",
  "confidence": "low | medium | high",
  "needs_adjudication": false,
  "short_rationale": "..."
}
```

All candidate pairs within each trace are compared. Judges do not see prior verdicts on other pairs.

D.4 Agreement and posterior diagnostics

Diagnostics (Table 11) are computed over the supplementary TriageBench traces (n=177 traces, mean 6.7 candidates per trace; 3,629 canonical pairwise comparisons). Pairwise winner agreement reaches Fleiss $\kappa = 0.646$ (substantial) and Krippendorff $\alpha = 0.594$; the 5-point ordinal α is 0.676. These council agreement coefficients are not directly comparable to Who&When's 9.1 to 50% disagreement on uncertain cases.

Quantity	Value
Per-trace BT strengths and posterior uncertainty	
BT-score variance, mean / med.	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 of 170 (92.9%)
Traces with top-1 post. > 0.70	118 of 170 (69.4%)
Traces with top-1 post. > 0.90	68 of 170 (40.0%)
Three-way council pairwise verdicts	
Cohen κ, GPT vs. Opus (3-cls)	0.638
Cohen κ, GPT vs. Grok (3-cls)	0.531
Cohen κ, 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 κ, 3 raters (3-cls)	0.646
Unanimous (complete cases)	1,228 of 1,658 (74.1%)

Table 11: TriageBench strength, posterior, and inter-annotator diagnostics. Council: GPT-5.4, Claude Opus 4.6, Grok 4.20. Strength and posterior rows use n=170 (traces with $\geq$ 2 fitted candidates; 7 of 177 excluded). Variance and spread summarize candidate MAP strengths within each trace. The 3-class scale is A / tie / B, collapsing preference strength. Complete cases require usable votes from all three judges. QWK = quadratic-weighted Cohen κ on the 5-point pairwise scale.

Across the 170 eligible traces, the BT posterior is concentrated: 92.9% have a single candidate carrying $> 50\%$ of top-1 mass, and 40% have $> 90\%$ on a single step, indicating that the fitted preference model often distinguishes a leading candidate. Quadratic-weighted 5-point κ values (0.625 to 0.721) summarize agreement on the ordinal pairwise scale. Pairwise winner agreement is strongest between GPT-5.4 and Claude Opus 4.6 (κ=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 agree on the preferred candidate or on a tie in 74.1% of complete three-judge comparisons, they collectively disagree with the human gold on $\sim 60\%$ 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, +18 pp 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 candidate steps are individually judged viable repair targets by each LLM. At the trace level, 174 of 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. Within the council's candidate pools, the BT top-1 matches the human-annotated step in 40.7% of traces, and the human step appears within the top-3 in 68.9% and top-5 in 84.2%. Each pool contains the human-annotated step, so first-choice disagreement is not caused by its exclusion. The top-3 and top-5 results describe its position within this restricted pool.

D.5 TRIAGE comparison

First-choice agreement. We compare the saved Kimi top-1 predictions underlying Table 1 with the published TriageBench council top-1 steps. All 177 benchmark traces match prediction records by exact trace ID, retaining the AG/HC subset suffix, and their original decisive-error step IDs agree. We use the saved predictions without restricting or reranking them to the council's pool. Of these traces, 136 have a prediction inside the pool, 40 have a prediction outside it, and one has no prediction. We count the missing prediction as a non-match: agreement is 58 of 177 (32.8%) overall and 58 of 136 (42.6%) conditional on the prediction being in the pool. This statistic measures agreement of first choices, not executed repair success. The original 42.4% localization result remains 78 of 184 against Who&When's labels; the council comparison uses a different target and trace set.

Rank correlation. We reconstruct the pre-replay priority $\rho(i)I(i)$ from saved suspicion and structural-impact values in the same Kimi run, using the recorded position mixture and $I_{\text{graph}} = 0$ for candidates outside the impact gate, as in the saved implementation. Scores are available for 1,124 of the 1,184 council candidates (94.9%); 862 of these also have finite council Bradley–Terry scores. We compute Spearman correlation within each trace on the paired scores, assigning average ranks to ties, then average across the 168 traces with defined correlations. Eight traces have fewer than two paired scores and one has constant council scores; these are excluded. Missing council scores are not treated as zero. The mean correlation is 0.271 (95% CI: $[0.190, 0.352]$, from 10,000 trace-bootstrap resamples). This interval reflects variation across traces, with the council estimates held fixed. Excluding the single trace with a recovered final prediction gives 0.275 across 167 traces, leaving the association essentially unchanged. This analysis concerns the pre-replay priority score over shared candidates; it does not evaluate the final repair shortlist or executed recovery.

E Additional Experiments

E.1 Evaluable subset

Main-body localization rates use the full benchmark denominators (n=184 Who&When, n=148 TRAIL), counting unavailable predictions as misses. The Who&When formula ablation uses 176 retained traces with an annotated error in the candidate pool. TRAIL formula analyses use the 144 retained traces with gold-label rows. These subsets differ from the trace sets used for patch plausibility, which does not require an error-location label.

E.2 Random and formula baselines

We compare TRIAGE with random selection to test whether its ranking places annotated errors earlier in the list (Table 12), and with simpler scoring functions to examine the contribution of each ranking component (Table 15).

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

Random top-K (Table 12). The random baseline samples K steps uniformly without replace-

ment from the entire normalized trace and counts a hit if any sample matches an annotated error. We use Monte Carlo with 200 seeds per trace.

E.3 ρ-only suspicion ablation

The ρ-only row in Table 13 ranks steps using suspicion scores, without structural impact I or replay R. Relative to this row, adding structural impact gains +0.6pp top-1 on Who&When and +2.1pp on TRAIL; changes at intermediate K are small and do not consistently favor ρI.

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
ρ-only	41.8	**65.2**	**74.5**	82.6
TRIAGE (ρI)	**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
ρ-only	54.7	**78.4**	84.5	91.2
TRIAGE (ρI)	**56.8**	77.7	84.5	**93.2**

Table 13: LLM-as-judge vs. ρ-only vs. TRIAGE on Who&When and TRAIL top-K recall (%). target = 10 is a single-call ranked list capped at 10; ρ-only is TRIAGE's suspicion stage in isolation.

E.4 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 14 compares the rankings in Figure 2 on traces with retained gold labels (n=178 W&W, n=144 TRAIL), counting missing predictions as misses. On TRAIL, TRIAGE's top-3/5/10 advantages are statistically significant ($p = 0.049$, $p = 0.009$, and $p < 0.001$); top-1 is not significant (p=0.169). On Who&When, no K shows a statistically significant gap (p>0.18 across all K).

K	Δ (pp)	95% CI	p
Who&When (n=178 paired, single-error hit)			
1	+2.8	[−2.2, +7.9]	0.339
3	+1.1	[−4.5, +6.7]	0.773
5	+1.7	[−3.4, +6.8]	0.595
10	+3.4	[−1.1, +7.9]	0.189
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 14: 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.5 Formula ablation

Who&When. Table 15 reports the formula ablation on the Who&When evaluable subset (n=176). All rows rerank the same candidates; only the ranking formula changes.

ρ-only top-5 (77.8%) marginally exceeds $\rho \cdot I$ top-5 (77.3%); the gains from $\rho \cdot I$ over ρ-only are concentrated in W ± 2 (+0.6 pp) and MRR (+0.010), with NDCG@3 essentially tied.

Formula	T1	W2	T5	MRR	N@3
Position (π)	11.9	36.9	59.7	.336	.282
Suspicion (ρ)	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**

Table 15: 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.

Formula	T1	T3	T5	T10	MRR
Position (π)	0.7	67.4	75.7	91.0	.341
Suspicion (ρ)	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

Table 16: Formula ablation on the TRAIL evaluable subset (n=144, any-error hit). Best per column **bold**; second-best underlined.

TRAIL. Table 16 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 π adds +1.1pp top-1. The body reports these numbers at the full n=148 denominator (counting four traces without retained gold labels as misses): 54.7/78.4/84.5/91.2 for ρ-only, 56.8/77.7/84.5/93.2 for TRIAGE.

Uniform-weight ablation. We recompute I_{graph} on Who&When (n=178) with the family-pair weights (f_{ij}, Table 6) and per-family criticality (c^{type}, Table 7) set 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 weights (Table 17), while the other reported metrics show small, mixed differences. This comparison provides limited evidence for the particular weight choices.

Variant	T1	T3	T5	T10	W±2
ρ-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 17: Uniform-weight ablation on Who&When (n=178). TK = Top-K step accuracy (%); W±2 = within-2 accuracy.

F Cost Decomposition

Table 18 decomposes the Kimi K2.5 Who&When run by pipeline stage. With M=3, one suspicion pass, two patch-generation calls per candidate, and two replay calls per candidate require 13 calls per trace before retries; the P=5 configuration requires 17. The table includes recorded requests and repair attempts, so its total need not equal this nominal budget. Suspicion accounts for an estimated 44.1% of total cost; patch hypothesis and generation together account for 34.0%. The run records 534 baseline and 530 patched replay requests (1,064 in total).

Stage	Calls	In tok	Out tok	% cost
Normalize	0	0	0	0.0
Suspicion	890	7.700M	0.890M	44.1
Impact graph	0	0	0	0.0
Patch hypoth.	531	4.711M	0.080M	11.2
Patch gen. + repair	980	7.720M	0.720M	22.8
Baseline replay	534	4.249M	0.119M	11.4
Patched replay	530	4.023M	0.108M	10.5
Total	3,465	28.403M	1.917M	100.0

Table 18: Estimated cost decomposition for Kimi K2.5 on Who&When (n=178, ∼$15.53 total). Rates per million tokens: $0.60 uncached input, $0.15 cached input, $2.50 output. Suspicion uses the retained token estimate without a cache discount; missing replay usage uses stage means.

G Human + LLM-Panel Patch Rating Study

Motivation. Localization accuracy does not directly measure the usefulness of generated repairs. We complement it with a separate rating study of diagnostic outputs.

Study design. The study retains the ranked candidates and repairs from an earlier diagnostic configuration. Sample selection, rating questions, and the analysis plan were fixed before rating. **(1) Sample**: all 326 traces in the full validation pool, covering 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. The pool contains 977 candidates across 326 traces. **(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 κ. All ratings are reported, including negative findings.

Raters. One human rater and three LLM judges (Claude Opus 4.6, GPT-5.4, Grok 4.20; the same models as the TriageBench council) rate candidates on Q1 and Q2 using the same rubric and access to the gold answer. The human pass took roughly 35 hours over multiple sessions. LLM judges rate each candidate independently. These are individual candidate ratings (good / ok / poor / unsure), rather than pairwise comparisons.

Results. Table 19 reports rating distributions per rater per question over the available ratings. 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%).

Rater	Q	n	good	ok	poor	unsure	good+ok
Human	step	977	38.1	51.5	9.1	1.3	89.6
Human	patch	977	24.4	54.6	19.7	1.4	78.9
Opus 4.6	step	968	43.4	40.5	16.1	0.0	83.9
Opus 4.6	patch	968	33.1	40.8	25.9	0.2	73.9
GPT-5.4	step	977	45.9	40.8	13.3	0.0	86.7
GPT-5.4	patch	977	36.5	35.6	27.4	0.4	72.2
Grok 4.20	step	977	58.8	35.6	5.6	0.0	94.4
Grok 4.20	patch	977	73.0	19.5	7.5	0.0	92.5

Table 19: Per-rater rating distributions (%); n is the number of rated candidates.

Per-trace headline rates. Table 20 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 strict top-1 is 52.1% (at least 2 of 3 judges rating *good*).

Rater	Q	n	T1$_g$	T1$_{g+o}$	A3$_g$	A3$_{g+o}$
Human	step	326	60.1	92.6	84.7	98.8
Human	patch	326	34.0	86.5	56.7	98.5
Opus	patch	323	44.3	85.1	69.7	96.3
GPT	patch	326	45.7	76.7	78.5	97.2
Grok	patch	326	82.5	95.7	99.1	100.0
Panel maj. (3)	patch	326	52.1	86.2	81.6	99.1

Table 20: Per-trace plausibility rates (%). T1 and A3 denote top-1 and top-3-any; subscripts g and $g+o$ denote good and good+ok. n counts rated traces. Panel majority requires at least two LLM judges to meet the threshold for the same patch.

Per-stratum patch plausibility. Table 21 reports per-stratum human rates. SWE-Bench has the lowest human patch-plausibility rates under each reported threshold.

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

Inter-rater agreement. Table 22 reports Cohen's κ between the human rater and each LLM judge. Opus and GPT agree more closely with the human rater than Grok does. Grok assigns substantially more patches a *good* rating (73.0% vs. 24.4% for the human), illustrating the sensitivity of plausibility ratings to the choice of judge.

	4-class κ		Binary κ	
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 22: Cohen's κ 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*).

Scope of the study. The study contains 968 to 977 candidate ratings per question and rater, with 31 to 123 traces per stratum. One human rater cannot establish broader practitioner agreement, and the LLM panel provides additional judgments rather than independent human validation. Access to the gold answer may also affect plausibility ratings relative to a deployment setting where the answer is unknown. These ratings assess perceived plausibility; executed interventions are needed to establish repair effectiveness.

H Reproducibility

Table 23 lists the exact model IDs. Pipeline settings and prompts are detailed in Section 5.4 and Appendices A–C; costs are reported in Appendix F. The TriageBench release includes trace-level rankings and pairwise judgments.

Provider	Model ID
TRIAGE pipeline	
Moonshot	`kimi-k2.5`
Anthropic	`claude-sonnet-4-20250514`
Together	`meta-llama/Llama-3.3-70B-Instruct-Turbo`
TriageBench council and patch-rating panel	
OpenAI	`gpt-5.4-2026-03-05`
Anthropic	`claude-opus-4-6`
xAI	`grok-4.20-0309-reasoning`

Table 23: Model identifiers.