MAKING EVIDENCE COUNT: GROUNDED DIAGNOSIS FOR SOFTWARE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diagnosing a software failure requires more than locating relevant code. An agent must connect the failure mechanism to the observed symptoms, identify where and how a repair should occur, and ground these claims in acquired evidence. We make three contributions toward evaluating and supporting this reasoning. First, we introduce FailureSnapshotHub (FSH), an interactive benchmark of 60 real continuous-integration failures from 49 repositories. Agents investigate only failed-side logs, code, tests, configuration, and execution context, while expert-adjudicated references and corroborating developer repair diffs remain hidden. Evaluation separately assesses mechanism, causal direction, repair owner/location, repair direction, completeness, and evidence grounding. Second, we formulate evidence-centric context compilation and present EviCompile, a deterministic method that transforms tool observations into source-linked evidence memory through recoverable text coding, typed evidence construction, cross-observation consolidation, and cost-aware global ordering. EviCompile itself neither generates new observations nor invokes another language model; causal inference remains with the diagnostic agent. Third, we empirically characterize the effects of evidence representation in a sequential Tool Agent, an OpenRCA-style controller–executor agent, and fixed-evidence diagnosis. Case analyses show that agents can locate the relevant code or recognize an immediate mismatch yet infer the wrong mechanism or repair boundary. Compared with original observation histories, EviCompile improves mean semantic scores by 9.72 and 10.42 points in the two interactive settings and by 5.55 points when acquired evidence is held fixed. Local text coding underperforms EviCompile in all three settings; with evidence membership fixed, full ordering also outperforms chronological and relevance-only presentation. Blinded expert reassessment on 40 cases preserves positive mean gains. Together, these results show that evidence organization can improve causal and repair judgments without additional observations, while FSH exposes diagnostic errors that location-based evaluation cannot resolve.

1 INTRODUCTION

Fault localization identifies where an investigation should focus. Diagnosis must explain how a defect produces the observed failure and where a corrective change belongs. A failed assertion can expose an invalid value produced upstream: the assertion and its producer have different causal roles and repair responsibilities. Even recognizing a violated interface contract does not establish which side should change. We study how to evaluate these distinctions and represent evidence so that software agents can reason about them.

We introduce **FailureSnapshotHub (FSH)** to define both the investigation environment and the explanatory obligations. Its 60 BugSwarm-derived cases span 49 repositories and expose failed-side logs, code, tests, configuration, and execution context through a common tool interface. Experts screen for sufficient failed-side evidence and clear repair responsibility, then independently annotate and adjudicate references using those materials and the historical developer repair diff. The diff corroborates the intervention but cannot supply an otherwise missing diagnostic premise. Neither diffs nor references are accessible to agents. Evaluation separates mechanism, causal direction, repair owner/location, repair direction, completeness, and evidence grounding.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

These distinctions matter in practice. In splink, an agent names the correct function but attributes a SQL failure to the wrong construct. In OneBusAway, an answer identifies a mapping/API mismatch yet assigns repair to the mapping rather than the model-and-caller migration supported by the reference. A location match cannot distinguish these diagnoses (Section 6; Appendix F).

Agents interleave reasoning and information gathering (Yao et al., 2023), but chronological tool histories can separate complementary facts and repeat symptoms. Local coding leaves repetition across responses; independent summaries may discard details before their relevance becomes apparent. Findings on input position (Liu et al., 2024) and agent interfaces (Yang et al., 2024; 2025) motivate examining this representation problem.

In the parser example developed in Appendix C, a valid YAML request fails because a registry omits an existing handler. Contract and source evidence explain the missing registration, while log and stack tools repeatedly expose the same exception. The agent must recover the causal relationship from this fragmented presentation. This motivates changing the organizing unit of context from individual tool responses to source-linked evidence.

We introduce **EviCompile**, which deterministically compiles observations into typed, source-linked records, shares compatible content while retaining acquisition attribution, and orders the complete memory by relevance, complementary coverage, and representativeness. It neither acquires observations nor invokes another language model; causal inference remains with the diagnostic agent. Figure 1 summarizes the benchmark, compiler, and assessment.

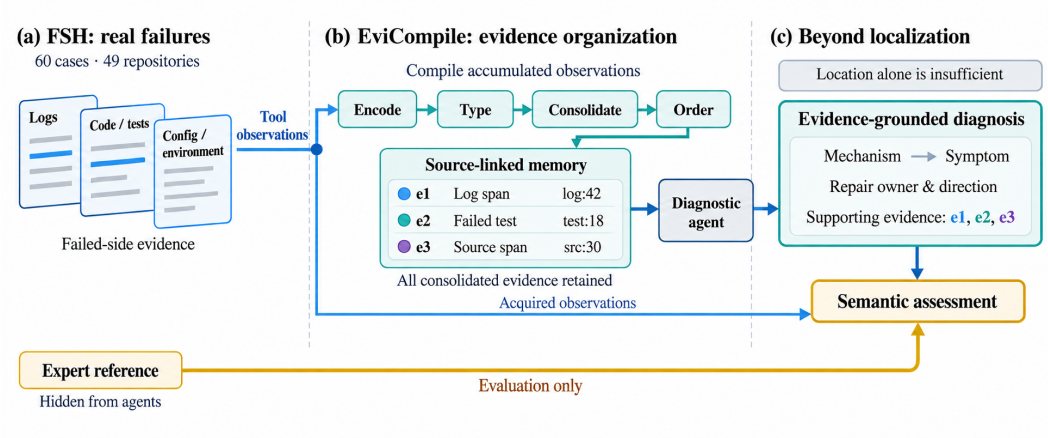

Figure 1: Overview of evidence-grounded diagnosis. (a) FSH exposes failed-side materials for 60 CI failures from 49 repositories; repair diffs and expert references remain hidden from agents. (b) EviCompile deterministically compiles tool observations into source-linked memory, retaining all consolidated evidence. (c) The agent explains mechanism and causal progression, identifies repair owner and direction, and cites evidence. Assessment uses the diagnosis, acquired-observation replay, and private reference. Examples are schematic; control loops and baselines are omitted.

Our contributions are: **(1)** an interactive diagnosis benchmark with expert screening, separated agent/reference materials, and six-dimensional evaluation beyond localization; **(2)** an evidence-centric context compiler combining recoverable coding, typed records, cross-observation sharing, and complete cost-aware ordering; and **(3)** a controlled study across two interactive settings and fixed-evidence diagnosis, supported by an ordering ablation, real-case analyses, and blinded expert reassessment. The benchmark and implementation will be made publicly available soon.

2 Related Work

Software failures and diagnostic benchmarks. BugSwarm (Tomassi et al., 2019) provides reproducible failing/passing CI artifacts. Our FSH benchmark adds an interactive investigation protocol and expert-adjudicated diagnostic references. SWE-bench (Jimenez et al., 2024) instead evaluates repository modifications that resolve real issues.

Historical test logs have been mined to predict bugs and localize faults, while spectrum-based log diagnosis identifies events associated with failures (Amar & Rigby, 2019; Rosenberg & Moonen, 2020). At the system level, work on AIOps benchmark construction and end-to-end troubleshooting combines logs, metrics, traces, and service topology to support root-cause localization (Li et al., 2022; Lee et al., 2023).

OpenRCA (Xu et al., 2025) studies enterprise-telemetry RCA with a controller–executor agent; OpenRCA 2.0 (Fang et al., 2026) adds fault-injection-based causal supervision. Our FSH benchmark addresses CI failures, evaluating mechanisms, causal progression, repair responsibility, and evidence grounding. Agents see failed-side snapshots; private references combine expert analysis with corroborating developer diffs.

Complementary work studies language-model capabilities for code execution reasoning and debugging, including self-debugging and repository-level repair (Chen et al., 2024; Ni et al., 2024; Xia et al., 2024).

Agent interfaces and observation design. Agent behavior can also be structured through planning procedures and external diagnostic knowledge. AdaPlanner and ADaPT adapt or decompose plans from interaction feedback, while FlowBench evaluates workflow-guided planning (Sun et al., 2023; Prasad et al., 2024; Xiao et al., 2024). In RCA, Flow-of-Action encodes standard operating procedures, in-context methods retrieve historical incidents, and Cloud Atlas derives causal guidance from system documentation (Pei et al., 2025; Zhang et al., 2024; Xie et al., 2024).

ReAct (Yao et al., 2023) interleaves reasoning and actions; SWE-agent (Yang et al., 2024) and AgentOccam (Yang et al., 2025) tailor agent interfaces. EviCompile organizes persistent evidence *after* tools return, without replacing planners or adding agent roles.

Prompt compression and long-context use. LLMLingua (Jiang et al., 2023) compresses prompts; LongLLMLingua (Jiang et al., 2024) targets long-context information density and position; LLMLingua-2 (Pan et al., 2024) learns token selection through distillation. EviCompile combines typed records, cross-observation content sharing, and persistent global ordering.

3 FailureSnapshotHub: Evidence-Grounded Diagnosis

3.1 Task and investigation boundary

An instance comprises an initial description q, snapshot $\mathcal{X}$, allowed tools $\mathcal{T}$, and private reference g. At step t, an agent chooses $a_t \in \mathcal{T}$ and receives o_t, yielding

$$H_t = \big(q, (a_1, o_1), \ldots, (a_t, o_t)\big). \tag{1}$$

It may continue investigating or return a diagnosis explaining the mechanism and causal progression, specifying the repair owner and direction, and supporting its claims with observed evidence. *Repair responsibility* denotes the component or boundary where a change addresses the mechanism, not necessarily the last stack frame. Equivalent implementations at that boundary are acceptable without reproducing the historical edits.

We construct FSH from the failed–passed CI artifacts provided by BugSwarm (Tomassi et al., 2019). Artifact screening requires a usable pair and sufficient repository, build, reproduction, and failure metadata. Construction can inspect both sides, but runtime policy restricts agents to the failed-side projection: logs, source, tests, configuration, environment facts, and bounded resource summaries. Passing artifacts, patches, references, evaluator labels, and answer-bearing metadata are excluded. Addressable log signals and source references support bounded reads and searches; indices guide access without labeling root causes. Appendix H details construction and coverage.

3.2 Expert screening and reference construction

Experts require sufficient failed-side evidence to infer mechanism, causal progression, and repair responsibility without the repair diff. They also reject cases with multiple independently viable

alternative repair locations consistent with intended behavior. A supported intervention may still coordinate several files, and semantically equivalent implementations remain valid.

Three experts independently annotate root-cause concepts, repair intervention, causal chain, and supporting evidence, then adjudicate field-wise disagreements. They use failed-side materials and the developer diff, which checks agreement with the historical correction; model answers and evaluation outputs are not annotation inputs. The reference is a set of diagnostic concepts and obligations rather than one answer string. The diff remains hidden throughout agent investigation.

The screening criterion concerns competing alternatives, not the number of edited files. A case is excluded if essential diagnostic premises exist only in the repair, or if the historical diff merely selects among equally viable repair targets. Reference construction therefore uses the diff to corroborate a supported intervention rather than arbitrarily deciding responsibility. Causal roles separately describe how an invalid state is created, propagated, or revealed.

The benchmark contains 29 Python-dominated and 31 Java-dominated cases; all 60 have logs, environment artifacts, project files, and evidence indices. Table 1 describes the selected cases, not population frequencies. Appendix H gives category definitions and the source-extension grouping rule.

Table 1: Primary failure types in the selected 60-case benchmark. These are coverage categories, not population frequencies; percentages are independently rounded.

Primary failure type	**Count**	**Percentage**
Test oracle, harness, or external exposure	12	20.00%
Configuration, metadata, or template mismatch	10	16.67%
Generation, serialization, or query defect	10	16.67%
Implementation state or control defect	8	13.33%
Interface or API contract defect	7	11.67%
Runtime or dependency compatibility drift	7	11.67%
Missing or misplaced artifact	6	10.00%
Total	**60**	**100.00%**

3.3 Semantic assessment

FSH scores **mechanism, causal-chain direction, repair owner/location, repair-action direction, completeness, and evidence grounding**, each from 0 to 4. These assess why failure occurs, how effects reach the symptom, where and how to intervene, whether material obligations are covered, and whether acquired observations support the claims. Source-linked observation replay enables grounding checks. Section 5 defines aggregation; Appendix D gives scoring anchors. All representations use the same investigation interface and reference criteria.

4 EviCompile: From Tool History to Evidence Memory

4.1 Recoverable coding, typing, and sharing

Given accumulated observations $O_{1:t}$, EviCompile applies four deterministic operations (Figure 2):

$$\mathcal{P}_t = \mathrm{Order}_q\left(\mathrm{Share}\left(\bigcup_{i=1}^{t}\mathrm{Extract}\big(\mathrm{Code}(o_i)\big)\right)\right). \tag{2}$$

The initial task context remains separately available. $\mathcal{P}_t$ is an ordered memory, not a generated summary or an inferred root cause.

Code. Terminal normalization removes formatting noise; repetitive lines, timestamp prefixes, and numeric templates receive compact, recoverable codes. Coding applies only to designated text fields, protecting identifiers, source paths, locations, arguments, and schemas. A code is adopted only if shorter and if expansion preserves the normalized visible token sequence (Appendix A).

Type. Records contain type, content, available source location, role hint, and attribution. Types include exceptions, failed tests, source/log spans, environment signals, and path inventories. Role hints derive from observable structure, not private references. A residual-fact channel retains unrecognized scalar information (Appendix B).

Share. Content identity is separated from acquisition history. Exact duplicates and matching exception signatures share content while preserving contributing observations and source references. Compatible path inventories use stable union; a source span shares a covering record only when its contained text matches. Similarity alone does not authorize merging. The pool is reconstructed after new observations, allowing later source reads to subsume compatible narrow spans while retaining provenance.

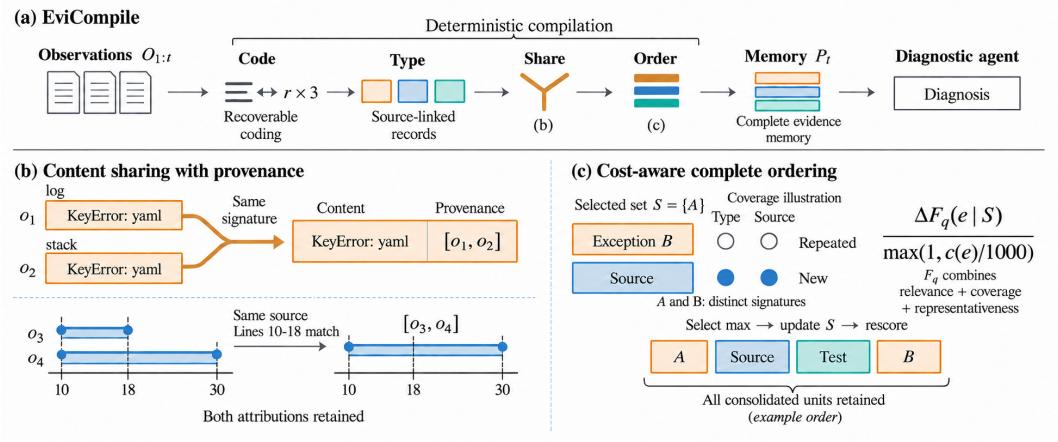

Figure 2: EviCompile transforms observations into evidence memory without an additional LLM. (a) Four deterministic stages precede agent reasoning. (b) Matching exception signatures and compatible containing source spans share content while retaining acquisition provenance. (c) Ordering repeatedly maximizes cost-normalized marginal utility, updates S, and rescores. F_q combines relevance, coverage, and representativeness; $c(e)$ is character cost. Hollow/filled circles indicate repeated/new type or source keys; A and B have distinct signatures. Examples are schematic; all consolidated units are retained.

The common record granularity lets a stack-trace tool and a log-span tool expose the same exception despite different response schemas. Retaining both attributions distinguishes recurring evidence acquisition from independent diagnostic facts. Rebuilding the pool also allows later, more informative reads to replace compatible narrow records without retaining each earlier packet as another context block.

Text coding and evidence organization have different preservation targets. Coding recovers the visible, line-wise token sequence after terminal normalization, not terminal-control bytes. Consolidation retains representative content and source attribution, while the compact model rendering need not repeat every internal bookkeeping field. Scalar counts, Boolean configuration values, and other unrecognized fields remain available through residual facts. These preservation contracts distinguish changing the presentation of acquired evidence from discarding observations or generating an independent summary.

4.2 Complete Cost-Aware Ordering

Relevance alone can prioritize repeated symptoms over complementary source or configuration facts. Let $\mathcal{E}$ be the consolidated pool, $r_q(e) \geq 0$ its relevance, and $\mathcal{K}(e)$ the coverage keys: type, role, source, signature, owner candidates, and tags. Let $s(u, e) \in [0, 1]$ be Jaccard similarity of content-token sets and $w_u \geq 0$ a representation weight. For a prefix set S,

$$F_q(S) = \sum_{e \in S} r_q(e) + \lambda \left| \bigcup_{e \in S} \mathcal{K}(e) \right| + \mu \sum_{u \in \mathcal{E}} w_u \max_{e \in S} s(u, e), \tag{3}$$

with maximum zero for an empty set. The terms capture relevance, complementary coverage, and representativeness. Relevance combines type/role priors, optional query matches, owner cues, and noise penalties. Starting with $S_0 = \varnothing$,

$$e_j = \arg\max_{e \in \mathcal{E} \setminus S_{j-1}} \frac{F_q(S_{j-1} \cup \{e\}) - F_q(S_{j-1})}{\max\{1, c(e)/1000\}}, \qquad S_j = S_{j-1} \cup \{e_j\}. \tag{4}$$

Here $c(e)$ is the rendered record's character cost. Lazy reevaluation updates gains after each selection; ties follow stable input order. **Every consolidated unit is retained:** the procedure produces a permutation, not a selected subset. For fixed nonnegative inputs, F_q is normalized, monotone, and submodular, explaining diminishing returns for repeated keys or patterns. This does not guarantee diagnostic correctness or an optimal prefix for every budget. Appendix A proves the property and gives the parameters ($\lambda = 0.25$, $\mu = 0.35$); Appendix J preserves the reconstruction algorithm.

4.3 Integration into diagnosis

In **Tool Agent**, each new observation triggers reconstruction of a global ledger used for decisions and final diagnosis; the action history retains lightweight evidence references. In **OpenRCA-style**, an action packet supports immediate executor interpretation, and an accumulated packet supports later executor stages and final diagnosis. The controller still receives executor summaries. In **Fixed-evidence**, the complete acquired collection is compiled once for a fresh diagnosis call, with no further tool use. Appendix B expands these interfaces.

5 Experimental Design

5.1 A crossed study with matched operating limits

We evaluate three representations on the same 60 cases. **Raw** uses the original structured observations returned by FSH, retaining their observation-wise history without optional text coding or evidence compilation. **Text-coded** (`text_only`) applies local recoverable text coding while retaining that history structure. **EviCompile** additionally types evidence, deduplicates across observations, and reorders the accumulated memory.

We cross them with a serial **Tool Agent** (`direct`), an **OpenRCA-style** controller–executor agent based on OpenRCA (Xu et al., 2025), and **Fixed-evidence** diagnosis. We adapt the OpenRCA control structure to our FSH investigation interface. The first two settings allow each representation to influence the agent's ongoing investigation. In Fixed-evidence, the initial context and tool observations from a completed OpenRCA-style Raw investigation form a frozen evidence bundle. Raw, Text-coded, and EviCompile each render this same bundle for a fresh diagnosis-only call; matching bundle digests verify that the source evidence is identical. The source agent's generated summaries and final answer are excluded, and the downstream model cannot request additional tools.

All diagnostic stages request `deepseek-v4-flash`, temperature 0, top-$p = 1$, and non-streaming generation. Within each interactive setting, representations share the model, fixed prompt family, 19-action catalog, and 20-step ceiling (Appendix E). Every EviCompile condition renders *all* ordered evidence units. Observations are not hard-truncated at 8,000 characters, and no representation-level rank cutoff removes evidence from the compiled memory. Appendix E gives the recorded observation setting, aliases, output allowance, prompt contracts, and termination details.

The interactive comparisons measure *within-setting end-to-end effects*: representation can change actions, stopping, and token use despite shared ceilings. Fixed-evidence compares how the three representations support diagnosis from the same already acquired observations, so representation cannot change which evidence is collected. Neither comparison imposes equal realized token consumption or establishes efficiency.

5.2 SEMANTIC SCORING AND HUMAN VALIDATION OF METHOD COMPARISONS

The judge is also `deepseek-v4-flash`, using `fsh-semantic-dimensional.v1`. Given the reference, candidate, and acquired-observation replay without method labels, it returns six schema-constrained 0–4 scores: mechanism, causal direction, repair owner/location, repair direction, completeness, and evidence grounding. For answer y_i, three judge requests are aggregated separately within each dimension:

$$\text{Score}(y_i) = \frac{25}{6} \sum_{k=1}^{6} \text{median}_{r\in\{1,2,3\}}\, d_{i,k,r}. \tag{5}$$

This is a semantic score, not a solved-case percentage; localization and cost remain auxiliary. Each dimension uses shared anchors: 4 is correct without material omission; 3 is substantially correct with secondary omissions; 2 mixes material correctness with errors or missing elements; 1 is mostly incorrect or symptom-level; and 0 is missing, irrelevant, contradictory, or reversed. Mentioning a correct file or term does not earn credit when the surrounding claim is wrong. The judge supplies dimension-specific reasons and evidence excerpts, with no independent overall-score judgment or hidden penalty. Parsing failures are evaluation errors rather than automatic zero-quality diagnoses.

Human reassessment covers 40 benchmark cases. Each reviewed answer is rated by three experts with both method labels and automated scores hidden, and the human score is the arithmetic mean of their ratings. Human and automated totals use the same six semantic dimensions, equal weights, and conversion to the 0–100 scale. Averaging across human experts is distinct from the automated judge's per-dimension median across three requests. Random/stratified selection is supplemented by oversampling repair-boundary and grounding ambiguities, large EviCompile–Raw differences, and unstable repeated judgments.

We assess automated–human score agreement using pooled Pearson and Spearman correlations, exact agreement, and within-one agreement. To test whether the method comparison is preserved under human judgment, we compare Raw and EviCompile on the common reviewed subset within each of the three diagnostic settings, using the same reviewed outputs under both evaluators. We define automated gain bias as the automated EviCompile–Raw gain minus the human gain: negative values indicate attenuation, whereas positive values indicate inflation. These agreement and gain analyses use the same 40-case human-review protocol. Appendix D gives the sampling scope, expert aggregation rule, pooled agreement statistics, and signed-discrepancy analysis.

The targeted review supports comparisons on this selected subset, not unbiased population estimates or significance claims.

Repetitions and pairing. Each case–condition score averages three diagnostic runs, separately from repeated judging; we then average across cases. Bootstrap intervals and wins/ties/losses use paired case-level differences, not individual judge requests (Appendix G).

6 RESULTS AND ANALYSIS

6.1 DIAGNOSTIC DISTINCTIONS AND REPRESENTATION EFFECTS

Across the nine conditions, the within-case score range has mean 44.31 and median 29.17; $24/60$ cases span at least 50 points. Repair direction, repair owner/location, and grounding vary most. These ranges mix architecture and representation changes, so they do not isolate EviCompile. In splink, correct localization coexists with the wrong SQL mechanism; in OneBusAway, a correct mismatch explanation coexists with the wrong repair boundary (Appendix F).

EviCompile has the highest mean in every setting (Table 2), improving on Raw by 9.72, 10.42, and 5.55 points, with more wins than losses. Text-coded underperforms Raw in both interactive settings and exceeds it by 2.08 points in Fixed-evidence; EviCompile exceeds Text-coded by 11.11, 11.11, and 3.47 points. This combined comparison adds typing, sharing, ordering, and memory reconstruction; the ablation below isolates ordering.

Table 2: Mean diagnosis scores over 60 cases and three runs per condition. EC: EviCompile. Paired-bootstrap intervals and wins/ties/losses (W/T/L) compare EC with Raw at case level. Deltas use unrounded means; bold denotes the highest row mean, not significance.

Setting	Raw	Text-coded	EC	Δ vs. Raw [CI]	W/T/L
Tool Agent	61.81	60.42	**71.53**	$+9.72$; $[6.25, 13.19]$	42/16/2
OpenRCA-style	59.72	59.03	**70.14**	$+10.42$; $[7.64, 15.28]$	44/15/1
Fixed-evidence	59.03	61.11	**64.58**	$+5.55$; $[3.47, 7.64]$	32/27/1

6.2 Human reassessment

Pooled automated–expert Pearson and Spearman correlations are 0.901 and 0.913; exact and within-one agreement on the 0–4 dimensions are 83.6% and 97.1% (Appendix D). On the common 40-case subset, expert scoring preserves positive EC–Raw mean gains in all settings (Table 3). Automated gain bias (Auto gain minus Human gain) is negative: -2.77, -1.40, and -2.78 points. Automated means are lower for both representations, with larger downward discrepancies for EviCompile (Table 12). Thus, automated evaluation attenuates the observed mean gains on this subset; it does not imply every answer or dimension is under-scored or that all evaluator bias is absent.

Table 3: Human and automated scores on the same 40-case subset (0–100). Human scores average three blinded experts. Gain is EC minus Raw; bias is Auto gain minus Human gain, calculated from displayed means. These are distinct from Table 2.

Setting	Human Raw	Human EC	Human gain	Auto Raw	Auto EC	Auto gain	Bias
Tool Agent	63.89	75.69	+11.80	58.33	67.36	+9.03	-2.77
OpenRCA-style	63.19	75.00	+11.81	59.03	69.44	+10.41	-1.40
Fixed-evidence	65.28	72.22	+6.94	55.56	59.72	+4.16	-2.78

6.3 Ordering with evidence membership fixed

After acquisition, typing, and deduplication, all ordering conditions use the same complete evidence set (Table 4). FullOrdering adds 2.77 mean points over chronological presentation and 2.08 over relevance-only ranking. Simple relevance is 0.69 points above chronological order, but does not reproduce the full criterion's gain. FullOrdering is the Fixed-evidence EviCompile condition in Table 2; chronological order is not Raw because typing and sharing already apply.

Table 4: Fixed-evidence ordering ablation with identical evidence membership. FullOrdering matches EviCompile in Table 2. Bold denotes the highest mean and median, not significance.

Ordering	Mean	Median
TypedDedup-Chronological	61.81	67.36
TypedDedup-SimpleRelevance	62.50	68.75
TypedDedup-FullOrdering	**64.58**	**72.22**

6.4 Heterogeneous gains and repair responsibility

For cases with Raw score below 50, causal-chain direction has the largest listed dimension gain in every setting (Table 5); interactive gains also involve grounding, completeness, and ownership. The subset is retrospective and subject to headroom, baseline selection, and regression to the mean, not a deployable selection rule. The smaller fixed-evidence gap is consistent with Raw being more competitive on completed bundles, but subtracting effects across protocols would not isolate planning.

In **OneBusAway**, `GtfsMapping.hibernate.xml` persists `LocationGroup` through `locations`, while the model exposes `getStops()` and `addStop()`. Hibernate cannot find the corresponding JavaBean property. The reference identifies an incomplete model/API transition:

Table 5: Exploratory Raw-below-50 subset. Total-score gains use the 0–100 scale; dimension gains use 0–4. The subset is defined after evaluation.

Setting	n	Mean Δ	Median Δ	Largest reported dimension gains
Tool Agent	22	$+17.11$	$+11.11$	Causal chain $+2.045$; evidence $+1.909$
OpenRCA-style	24	$+20.37$	$+10.42$	Causal chain $+2.292$; completeness $+1.000$; owner $+0.875$
Fixed-evidence	16	$+9.03$	$+6.25$	Causal chain $+0.938$; mechanism $+0.729$

expose `getLocations()`, `setLocations()`, and `addLocation()` and update callers. Failed-side evidence supports this direction: `StopArea` already exposes the new API while the reader still uses `addStop()` for `LocationGroup`. The hidden developer diff corroborates that intervention.

The Tool Agent Raw answer earns $4/4$ for mechanism, causal direction, and grounding but proposes changing the mapping, receiving $0/4$ for repair owner and action. EviCompile raises these to $4/4$ and $3/4$, and the total from 58.33 to 87.50 (Table 6). A mapping-only change addresses the immediate incompatibility without recovering the reference's model-and-caller migration. Appendix F discusses alternative-repair interpretation and extends the analysis to instrumentation, test contracts, SQL, API migration, and prefix dispatch.

Table 6: OneBusAway diagnosis scores against the repair-informed reference. Correct mismatch recognition alone does not recover the model-and-caller repair boundary.

Setting	Raw	EviCompile
Tool Agent	58.33	87.50
OpenRCA-style	62.50	87.50
Fixed-evidence	62.50	79.17
Selected Tool Agent dimensions (0–4)		
Repair owner/location	0	4
Repair-action direction	0	3

7 Conclusion and Future Work

Successful diagnosis requires more than locating relevant code or recognizing an incompatibility. An answer can locate the failure yet explain the wrong mechanism, reverse causality, or assign repair to the wrong boundary. Diagnosis therefore requires relating evidence into a supported account of failure and repair.

FSH makes these requirements measurable through 60 interactive failed-side snapshots and expert-adjudicated, repair-informed references. Its six-dimensional assessment exposes errors in mechanism, causal reasoning, repair responsibility, completeness, and grounding that a location match can conceal.

EviCompile addresses the corresponding representation problem by compiling tool observations into typed, source-linked evidence memory with cross-observation consolidation and complete cost-aware ordering. Compared with Raw, it improves mean semantic scores by 9.72 and 10.42 points in the two interactive settings and by 5.55 points in Fixed-evidence diagnosis. Text-coded underperforms EviCompile in all three settings, indicating that local recoverable coding alone does not reproduce the full method's effect. With evidence acquisition, typing, and membership fixed, full ordering adds 2.77 points over chronological order and 2.08 points over relevance-only ordering. Blinded expert reassessment preserves positive EviCompile–Raw mean gains on the 40-case review subset. The Fixed-evidence result shows that reorganizing an already acquired evidence bundle can change the resulting diagnosis; the interactive differences are broader end-to-end effects and do not by themselves isolate a planning contribution.

Future work will evaluate EviCompile across additional models, languages, and agent architectures, and extend evidence-grounded diagnosis beyond CI failures to production incidents and other software-engineering settings.

AI USE STATEMENT

Generative AI assisted with manuscript drafting, revision, LaTeX typesetting, and figure-caption editing, literature retrieval, formal presentation of the evidence-ordering criterion and its proof, interpretation of the supplied aggregate results, and consistency checking of the experimental presentation. Language models are also used as the diagnostic agents and semantic evaluators described in the experiment. The expert reference diagnoses were independently annotated and adjudicated from failure-side materials and the corresponding developer repair diffs, without model answers or scoring outputs as annotation inputs. The authors are responsible for the final submitted text, claims, and artifacts.

ETHICS STATEMENT

The benchmark uses software-development artifacts and exposes a restricted failed-side investigation surface rather than unrestricted access to an operational system. Raw execution scripts and answer-side artifacts are excluded from agent access, and environment summaries are sanitized. Redistribution of source material remains subject to upstream licenses. An agent-generated diagnosis or proposed intervention is not an independently validated patch; deployment decisions require the checks appropriate to the target software system.

REPRODUCIBILITY STATEMENT

Section 3 specifies expert screening for diagnostic sufficiency and repair-responsibility ambiguity, distinguishes agent-visible investigation snapshots from reference construction using failure-side materials and developer repair diffs, and Section 4 specifies the evidence-compilation method. Appendix A gives the ordering objective, assumptions, proof, and parameterization; Appendix B describes evidence records and full-context rendering. Section 5 and Appendices D, E, and G document model and decoding settings, investigation limits, prompt contracts, semantic scoring, blinded three-expert reassessment, pooled score agreement, mean-score discrepancies, gain preservation, and case-level aggregation. Appendix E maps the paper's representation labels to the recorded experiment identifiers, and Appendix F provides the real-case analyses supporting the interpretation of the results. Appendix H documents screening metadata, construction records, failure-type definitions, snapshot characterization, and case-list and gold-reference identifiers; Appendix I states the ordering-ablation controls and interpretation.

Code and benchmark availability. The FailureSnapshotHub (FSH) benchmark and the EviCompile implementation will be made publicly available soon.

REFERENCES

Anunay Amar and Peter C. Rigby. Mining historical test logs to predict bugs and localize faults in the test logs. In *Proceedings of the 41st International Conference on Software Engineering*, pp. 140–151, 2019. doi: 10.1109/ICSE.2019.00031. URL https://doi.org/10.1109/ICSE.2019.00031.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KuPixIqPiq.

Aoyang Fang, Yifan Yang, Jin'ao Shang, Qisheng Lu, Junjielung Xu, Rui Wang, Songhan Zhang, Yuzhong Zhang, Boxi Yu, and Pinjia He. OpenRCA 2.0: From outcome labels to causal process supervision. arXiv preprint arXiv:2606.27154, 2026. URL https://arxiv.org/abs/2606.27154.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, 2023. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825/.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024. doi: 10.18653/v1/2024.acl-long.91. URL https://aclanthology.org/2024.acl-long.91/.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.06770.

Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, and Michael R. Lyu. Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. In *Proceedings of the 45th International Conference on Software Engineering*, pp. 1750–1762, 2023. doi: 10.1109/ICSE48619.2023.00150. URL https://arxiv.org/abs/2302.05092.

Zeyan Li, Nengwen Zhao, Shenglin Zhang, Yongqian Sun, Pengfei Chen, Xidao Wen, Minghua Ma, and Dan Pei. Constructing large-scale real-world benchmark datasets for AIOps. arXiv preprint arXiv:2208.03938, 2022. URL https://arxiv.org/abs/2208.03938.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Ansong Ni, Miltiadis Allamanis, Arman Cohan, Yinlin Deng, Kensen Shi, Charles Sutton, and Pengcheng Yin. NExT: Teaching large language models to reason about code execution. arXiv preprint arXiv:2404.14662, 2024. URL https://arxiv.org/abs/2404.14662.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024. doi: 10.18653/v1/2024.findings-acl.57. URL https://aclanthology.org/2024.findings-acl.57/.

Changhua Pei, Zexin Wang, Fengrui Liu, Zeyan Li, Yang Liu, Xiao He, Rong Kang, Tieying Zhang, Jianjun Chen, Jianhui Li, Gaogang Xie, and Dan Pei. Flow-of-Action: SOP enhanced LLM-based multi-agent system for root cause analysis. In *Companion Proceedings of the ACM Web Conference 2025*, pp. 422–431, 2025. doi: 10.1145/3701716.3715225. URL https://doi.org/10.1145/3701716.3715225.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. ADaPT: As-needed decomposition and planning with language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 4226–4252, 2024. doi: 10.18653/v1/2024.findings-naacl.264. URL https://aclanthology.org/2024.findings-naacl.264/.

Carl Martin Rosenberg and Leon Moonen. Spectrum-based log diagnosis. In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, 2020. doi: 10.1145/3382494.3410684. URL https://arxiv.org/abs/2008.06948.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. AdaPlanner: Adaptive planning from feedback with language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.16653.

David A. Tomassi, Naji Dmeiri, Yichen Wang, Antara Bhowmick, Yen-Chuan Liu, Premkumar Devanbu, Bogdan Vasilescu, and Cindy Rubio-González. BugSwarm: Mining and continuously growing a dataset of reproducible failures and fixes. In *Proceedings of the 41st International Conference on Software Engineering*, pp. 339–349, 2019. doi: 10.1109/ICSE.2019.00048. URL https://doi.org/10.1109/ICSE.2019.00048.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based software engineering agents. arXiv preprint arXiv:2407.01489, 2024. URL https://arxiv.org/abs/2407.01489.

Ruixuan Xiao, Wentao Ma, Ke Wang, Yuchuan Wu, Junbo Zhao, Haobo Wang, Fei Huang, and Yongbin Li. FlowBench: Revisiting and benchmarking workflow-guided planning for LLM-based agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10883–10900, 2024. doi: 10.18653/v1/2024.findings-emnlp.638. URL https://aclanthology.org/2024.findings-emnlp.638/.

Zhiqiang Xie, Yujia Zheng, Lizi Ottens, Kun Zhang, Christos Kozyrakis, and Jonathan Mace. Cloud Atlas: Efficient fault localization for cloud systems using language models and causal insight. arXiv preprint arXiv:2407.08694, 2024. URL https://arxiv.org/abs/2407.08694.

Junjielong Xu, Qinan Zhang, Zhiqing Zhong, Shilin He, Chaoyun Zhang, Qingwei Lin, Dan Pei, Pinjia He, Dongmei Zhang, and Qi Zhang. OpenRCA: Can large language models locate the root cause of software failures? In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=M4qNIzQYpd.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.15793.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. AgentOccam: A simple yet strong baseline for LLM-based web agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f2c6e459b95694a24ac69c469a4ee746-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Xuchao Zhang, Supriyo Ghosh, Chetan Bansal, Rujia Wang, Minghua Ma, Yu Kang, and Saravan Rajmohan. Automated root causing of cloud incidents using in-context learning with GPT-4. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pp. 266–277, 2024. doi: 10.1145/3663529.3663846. URL https://doi.org/10.1145/3663529.3663846.