SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

AI-Driven Root Cause Analysis of Production Incidents in Microservice-Based Cloud Applications

Vamsi Krishna Bhavana[1], Rohith Balerao[2]

[1]Application Support Engineer, Customer Success, Cresta, United States. E-mail: vamsikrishnabhavana5@gmail.com, ORCID: 0009-0006-4682-8593 (https://orcid.org/0009-0006-4682-8593)
[2]Senior HRIS Analyst, Chicago, Illinois, USA. E-mail: rohith.balerao23@gmail.com, ORCID: 0009-0002-3260-9323 (https://orcid.org/0009-0002-3260-9323)

Abstract
Production incidents in microservice-based cloud applications are difficult to diagnose because user-visible failures emerge from interactions among services, infrastructure, networks, deployment changes, and human operating practices. The same initiating fault may produce different symptoms across releases, while a single symptom may have several plausible causes. Existing artificial-intelligence approaches have made substantial progress: anomaly detection, service-dependency graphs, causal discovery, multimodal telemetry fusion, and large language models have each advanced the field along different fronts. Yet the literature remains fragmented. Many methods are evaluated on a small number of systems or fault types; reported performance depends on telemetry completeness and the definition of "root cause"; causal graphs may encode temporal association rather than interventionally valid causation; and rank-only recommendations can promote automation bias during time-pressured incident response. This paper synthesizes research on metrics-, log-, trace-, graph-, causal-, multimodal-, and language-model-based diagnosis and proposes **HITL-CausalFusion**, an uncertainty-aware, human-in-the-loop framework for production root cause analysis. The framework aligns heterogeneous telemetry and change events, learns modality-specific representations, and constrains temporal causal discovery with observed topology. It then ranks candidate causes using anomaly, propagation, counterfactual, and prior-knowledge evidence, and produces calibrated explanations with an explicit abstention mechanism. A rigorous evaluation protocol is specified across RCAEval, DeathStarBench-derived testbeds, OpenRCA, and controlled fault injection, using localization accuracy, ranking quality, calibration, robustness, latency, and operational utility. A complementary human study measures diagnostic correctness, time to a defensible hypothesis, workload, trust calibration, and unsafe reliance. The central argument is that publishable progress in AI-driven incident diagnosis should be judged not only by top-*k* localization, but by causal validity, transfer robustness, evidence traceability, calibrated uncertainty, and appropriate human reliance. The proposed framework and protocol provide a falsifiable foundation for such evaluation without overstating the maturity of autonomous root cause analysis.

Keywords: AIOps; root cause analysis; microservices; cloud computing; observability; causal inference; multimodal learning; incident response; explainable AI; human-in-the-loop

1. Introduction

Decomposing an application into independently deployable microservices buys operational flexibility, but it comes at a cost: a dense web of runtime dependencies now sits underneath that flexibility. A single user request might pass through gateways, a service mesh, databases, queues, caches, and third-party APIs before it returns, and because that chain crosses component and infrastructure boundaries, a failure anywhere along it can propagate well beyond its origin before an alert ever fires. Industrial and empirical studies describe incident response less as a hunt for the largest anomaly and more as a process of piecing together an evolving failure from incomplete evidence [1]–[4]. Tail latency, retries, backpressure, resource contention, and configuration changes can each turn a small, local defect into a symptom pattern spread across the whole system.

Root cause analysis (RCA), as used here, means identifying the initiating component, event, resource, or condition whose correction would prevent or substantially reduce the incident under investigation — a stronger, counterfactual claim than simply pointing at a correlated anomaly. In practice, though, the label attached to a root cause varies widely: it might name a service, a specific instance, a metric, a code change, a configuration value, or some combination of contributing conditions. This is part of why the literature is hard to compare. Work published under the single umbrella term RCA actually targets different endpoints — service localization, metric localization, failure-type classification, and narrative recommendation among them [1], [5] — and any comparison across these studies is misleading unless the diagnostic target and level of granularity are made explicit.

Three technical signals make up the core of observability: metrics summarize measurements over time, logs capture discrete events and state, and distributed traces connect operations along the path of a single request [6], [7]. Around these sit deployment, feature-flag, configuration, ownership, topology, and incident-history data, which add context the raw signals lack on their own. None of these modalities is complete by itself — metrics are compact but lossy, logs are expressive but inconsistent across services, and traces preserve request structure yet are often sampled rather than exhaustive. The harder problem is therefore not collecting more data, but aligning evidence that arrives on different clocks, at different scopes, under different sampling policies, and against different notions of what a failure looks like.

Several distinct families of methods have carried the RCA literature forward. Graph-based systems lean on dependency structure [8]–[11]; trace analysis picks out abnormal paths or operations [12], [13]; probabilistic and causal systems model how failures propagate or respond to interventions [14]–[17]; multimodal systems fuse logs, metrics, and traces [18]–[21]; and large language models retrieve, summarize, or reason over telemetry and incident histories [22]–[25]. Individually, the reported numbers can look striking: MicroRCA reported 89% precision and 97% mean average precision on its evaluated Kubernetes benchmark [9], and MicroHECL reported a top-three hit ratio of 68% with substantial reductions in localization time in an Alibaba deployment [10]. But numbers like these establish feasibility within a narrow evaluation, not universality — a point OpenRCA makes starkly, showing that even an agent built on a leading model solved only 11.34% of 335 enterprise failures in its benchmark [25].

This tension motivates a shift from "Which model has the highest accuracy?" to "Under what observability, causal, and organizational conditions is an AI recommendation reliable enough to guide an incident commander?" Production deployment requires uncertainty, missing-data behavior, evidence provenance, and human reliance to be first-class design variables, not afterthoughts bolted on once a model works in the lab. An incorrect high-confidence recommendation may waste the most valuable minutes of an incident; an unexplained correct recommendation may be rejected; and a useful system may fail organizationally if it is perceived as workforce surveillance or as a mechanism for assigning blame.

This paper makes four contributions:

1. It synthesizes the principal technical paradigms for AI-driven RCA and reconciles apparently conflicting findings by distinguishing system scope, telemetry assumptions, diagnostic granularity, and evaluation regime.
2. It proposes HITL-CausalFusion, a topology-constrained, temporally aware, multimodal framework that combines candidate ranking, counterfactual checking, uncertainty calibration, explanation, and human feedback.
3. It specifies a reproducible technical evaluation across benchmark, controlled, and production-like settings, including transfer, missing-modality, topology-drift, and open-set tests.
4. It defines a human-factors and governance protocol that measures workload, trust calibration, unsafe reliance, and equitable organizational adoption rather than treating acceptance as a secondary usability issue.

This paper is an integrative review and a conceptual framework, not an implementation report — it makes no claim to results that were never run. Everything said here about the proposed method is a hypothesis, and Section 7 lays out the protocol by which those hypotheses should be tested.

2. Background and Problem Definition

2.1 Production incidents as distributed, time-varying phenomena

A microservice application can be described at time t as a typed graph $G_t=(V_t,E_t)$. Its nodes span service instances, logical services, hosts, containers, data stores, queues, and external dependencies, while its edges capture request calls, message flows, deployment relationships, co-location, shared-resource relationships, and ownership. Neither set is fixed: autoscaling, failover, deployment, and orchestration all reshape nodes and edges continuously. A static dependency graph is therefore best treated as an imperfect prior, not a complete causal map.

Incidents don't all move at the same speed. A deployment can plant a latent defect hours before load exposes it; a memory leak grows slowly; a network interruption triggers an abrupt cascade; and retries can turn a downstream slowdown into upstream saturation almost immediately. None of this guarantees that the alert time matches the fault-onset time. Clock skew, aggregation windows, and sampling can all scramble the apparent order of events. Any method that leans on temporal precedence has to account for that uncertainty rather than assume the timeline it observes is the timeline that actually happened.

Operational RCA and retrospective postmortem analysis are not the same task. In the middle of an incident, engineers need the next safe test that will discriminate between hypotheses, not a final, tidy label. Once the

system has recovered, the organization's needs shift toward a defensible causal narrative, a record of contributing conditions, and corrective actions. Google's postmortem guidance emphasizes timely, blameless learning built around structured action items, and empirical studies of incident debugging back this up, showing that real practice combines tools, hypotheses, experience, and coordination rather than any single method [4], [26], [27]. A production RCA system needs to support both phases, and it should do so without flattening a complex incident down to one blame-bearing component.

A useful organizational design perspective also comes from outside the RCA literature proper: cross-domain research on enterprise agents. One prior study proposed a five-layer HRIS agent framework combining retrieval-augmented generation, multi-agent orchestration, explainable outputs, and human override [28]. That work was evaluated in a simulated HRIS setting, not on production incidents, so it is cited here for a narrower purpose — as support for an auditable, human-overridable decision-support pattern, not as evidence about telemetry localization or causal accuracy.

2.2 Telemetry modalities and failure evidence

OpenTelemetry defines traces, metrics, logs, and context-propagating baggage as observability signals [7]. Their diagnostic contributions and limitations differ.

Modality	Primary evidence	RCA value	Principal limitations
Metrics	Rates, latency distributions, errors, saturation, resource use	Efficient anomaly timing and scope; service and infrastructure comparison	Aggregation hides request-level variation; thresholds drift; correlation is not causation
Logs	Events, exceptions, state transitions, identifiers, messages	Rich semantics and failure-type clues	Template drift, volume, inconsistent levels, missing correlation identifiers, sensitive content
Traces	Request spans, parent-child paths, operation latency and status	Dynamic dependency and propagation evidence	Sampling bias, broken context, instrumentation gaps, asynchronous-flow complexity
Change events	Deployments, configuration, feature flags, scaling, ownership	Candidate initiating events and priors	Incomplete inventories, weak timestamps, undocumented manual changes
Incident knowledge	Prior tickets, runbooks, postmortems, mitigations	Retrieval of precedent and discriminating tests	Stale knowledge, survivorship bias, access controls, organization-specific language

Dapper established influential design principles for low-overhead, application-transparent distributed tracing [6]. Drain showed that online log parsing can efficiently map unstructured messages to templates [29], and OmniAnomaly demonstrated deep multivariate time-series anomaly detection with interpretable dimensions [30]. These techniques are useful building blocks, but anomaly detection alone cannot determine whether an abnormal component initiated, propagated, or merely reflected the failure.

2.3 What counts as a root cause?

Let Y_t denote a user-visible failure indicator and C a candidate condition. A strong causal target asks whether intervening on C—while holding relevant context constant—would substantially change Y_t. Formally, a causal effect can be expressed as a contrast between $P(Y_t \mid do(C=c_1))$ and $P(Y_t \mid do(C=c_0))$. Production telemetry is observational, so this quantity is usually not identifiable without structural assumptions, natural experiments, fault injection, or controlled intervention.

The term “root cause” is consequently used here with three safeguards. First, the output is a ranked hypothesis, not an ontological assertion. Second, the system distinguishes initiating causes, propagation mechanisms, and contributing conditions. Third, confidence is bounded by observed evidence and causal assumptions. When evidence is insufficient, abstention is a valid output.

3. Review Method and Research Questions

This is a focused integrative review, and it does not claim the exhaustiveness of a registered systematic review. Sources were drawn from peer-reviewed systems, software-engineering, services-computing, data-mining, machine-learning, and human-factors venues, supplemented by official observability documentation, recognized benchmarks, and governance standards. Priority went to primary papers that define a method, report an industrial study, release a benchmark, or evaluate multiple competing approaches; preprints were used only where necessary and were never treated as stronger evidence than peer-reviewed work. The evidence cut-off was 3 September 2026.

The synthesis was organized around four research questions:

RQ1. What evidence and representation strategies enable useful RCA across metrics, logs, traces, topology, and change events?

RQ2. How can temporal and causal reasoning distinguish an initiating cause from propagated symptoms without overstating causal identification?

RQ3. Which evaluation conditions reveal whether an RCA system is robust to missing telemetry, topology drift, new fault types, and domain transfer?

RQ4. How should explanations, uncertainty, workload, and organizational safeguards be designed to support appropriate human reliance during incident response?

Methods are compared here along several axes: diagnostic target, modality, structural assumptions, supervision, evaluation setting, explanation, and limitations. Reported percentages are used sparingly, and each one is explicitly attributed, since datasets and endpoints differ enough across studies that the numbers are not directly comparable.

4. State of the Art

4.1 Dependency- and graph-based localization

Dependency graphs make an intuitive substrate for tracking fault propagation. Brandón et al. evaluated graph-based RCA across three service-oriented and microservice architectures containing 25–56 containers and reported a 19.41% effectiveness improvement over the comparison used in that study [8]. MicroRCA correlates response-time and resource anomalies on an attributed graph, and its authors reported 89% precision and 97% mean average precision under injected anomalies in a Kubernetes benchmark [9]. MicroHECL takes a more dynamic approach, constructing service call graphs on the fly, searching anomaly propagation chains, and correlating candidate causes; its large-scale deployment shows the practical value of narrowing the search space by topology [10]. AutoMAP, meanwhile, builds anomaly behavior graphs from multiple metric types and uses directed random walks to locate causes even as the system changes underneath it [11].

What unites these methods is a sound operational intuition — failure evidence should be read in relation to the request and resource structure around it. Where they run into trouble is when an observed dependency gets mistaken for a causal direction, when asynchronous or shared-resource edges go unrecorded, or when the topology itself is stale. Random-walk scores, in particular, can end up favoring central or highly anomalous services even when those services are simply victims of the failure rather than its source. Graph construction and time alignment, in other words, are part of the inference problem itself, not a neutral preprocessing step before the real analysis begins.

4.2 Trace-centric approaches

TraceRCA localizes root causes from traces at a scale that holds up in practice [12]. MicroRank takes a different tack: it separates normal from abnormal traces, applies a PageRank-based operation scorer, and combines graph ranking with spectrum analysis, reporting recall gains of 6–22% over selected baselines [13]. What trace-based methods buy you is the request path itself — they can tell a normal operation from an anomalous one even when the aggregate service metrics appear similar.

Tracing, though, is rarely complete in practice. Head-based sampling can drop rare failures entirely, tail-based policies introduce their own selection effects, and broken context propagation leaves the trace graph fragmented. Background jobs and message queues further complicate parent-child relationships, and a trace-only method still misses host contention, deployment intent, and log semantics — none of which show up in a span. None of this makes traces weak evidence; it just means their quality needs to be estimated explicitly rather than assumed.

4.3 Probabilistic and causal reasoning

Sage applies causal Bayesian networks and machine learning to performance debugging, reporting culprit identification above 93% in its evaluated clusters [14]. CloudRCA takes a broader view, combining key performance indicators, logs, and topology inside a knowledge-informed hierarchical Bayesian network, and its authors reported more than a 20% reduction in SRE failure-resolution time in their deployment context [15]. CausalRCA aims at fine-grained localization and reported an average AC@3 of 0.719, ahead of the

baselines it was compared against [16]. RUN folds in temporal information through neural Granger causal discovery paired with personalized PageRank [17].

These systems push RCA beyond raw correlation, but the word "causal" is doing different work in each of them. Granger causality only asks whether past values improve prediction — it doesn't by itself establish an intervention effect. Bayesian networks need structural adequacy and, often, acyclicity to hold. NOTEARS makes learning a directed acyclic graph tractable through a smooth acyclicity constraint amenable to continuous optimization [31], but production systems routinely violate its assumptions through feedback loops, hidden confounding, nonstationarity, and aggregation. And when Pham et al. compared nine causal discovery methods against 21 RCA methods, they found no universal winner, just trade-offs among effectiveness, efficiency, sensitivity, scale, and how realistic the synthetic data actually was [32].

The upshot is that causal structure should be treated as a hypothesis, constrained by runtime topology and operational knowledge, rather than as an established fact. Validating edges, where possible, calls for fault injection or safe interventions. A strong anomaly-to-failure association can still be useful even without that validation — it just needs to be labeled for what it actually is.

4.4 Multimodal learning

Eadro folds anomaly detection and RCA together using metrics, traces, and logs in a single end-to-end multitask framework [18]. DiagFusion goes further, learning event representations, augmenting scarce failure data, incorporating deployment and trace relationships, and applying graph neural networks; its authors reported substantial gains over selected baselines for both localization and failure-type diagnosis [19]. FAMOS instead uses modality-specific extractors with late fusion and attention, and its 2025 study reported a 20.33% F1 improvement over the state-of-the-art methods it was compared against [20]. HolisticRCA takes yet another route, standardizing heterogeneous observability data and reasoning across service dependencies [21].

Multimodal systems can bring complementary evidence to bear, but fusing that evidence naively can just as easily amplify noise. A missing trace is not the same thing as a normal trace, and a flood of secondary errors can drown out a single, sparse log event that actually initiated the failure. Modalities also don't behave consistently across services or over time. Fuse them too early and incompatible granularities get forced into one representation; fuse them too late and cross-modal interactions get lost. What a production design actually needs is reliability-aware gating, explicit handling of missingness, and tests that deliberately remove or corrupt each modality in turn.

4.5 Large language models and agents

Where LLMs earn their keep is summarizing incident context, retrieving prior cases, proposing candidate causes, and generating explanations a human can actually read. Ahmed et al. studied more than 40,000 incidents across over 1,000 Microsoft services, evaluating root-cause and mitigation recommendations under fine-tuned, zero-shot, and multitask formulations [22]. RCACopilot takes a more structured approach, routing alert types to diagnostic handlers, aggregating evidence, and generating a root-cause category with a narrative; it reported accuracy up to 0.766 in its evaluation [23]. AIOpsLab, for its part, offers an agent-evaluation framework that deploys environments, injects faults, executes workloads, exposes telemetry, and scores the resulting agents [24].

OpenRCA is the clearest cautionary tale here. Its 335 failures, drawn from three enterprise systems, come with more than 68 GB of logs, metrics, and traces attached, and even so, the best evaluated model-agent combination solved only 11.34% of cases [25]. Long contexts, heterogeneous evidence, system-specific semantics, and dependency reasoning all remain genuinely hard, and LLM explanations can be fluent without being faithful to what actually happened. For that reason, the proposed framework restricts LLMs to a controlled explanation and interaction layer, grounded in evidence the machine has already selected — the language model never invents telemetry, never executes remediation on its own, and never overrides a calibrated abstention.

4.6 Cross-study synthesis

Method family	Representative systems	Strength	Recurring validity risk
Graph/topology	Graph-RCA, MicroRCA, MicroHECL, AutoMAP	Narrows search using propagation structure	Stale/missing edges; centrality mistaken for cause
Trace-centric	TraceRCA, MicroRank	Request-level paths and anomalous operations	Sampling, context gaps, asynchronous workflows

Method family	Representative systems	Strength	Recurring validity risk
Probabilistic/causal	Sage, CloudRCA, CausalRCA, RUN	Temporal and structural hypotheses	Hidden confounding; causal language exceeds identification
Multimodal	Eadro, DiagFusion, FAMOS, HolisticRCA	Complementary semantics and coverage	Noisy fusion; missingness; domain shift
LLM/agent	Incident recommendation, RCACopilot, AIOpsLab, OpenRCA	Retrieval, interaction, narrative synthesis	Hallucination; context limits; unfaithful confidence

Once evaluation scope is taken into account, these findings stop looking contradictory. Specialized methods can do very well on known services, fault injections, or fixed diagnostic labels. It's cross-dataset and open-ended benchmarks that impose the harder conditions — unseen faults, incomplete context, longer telemetry, shifting topology, and less constrained answers. The right research objective, then, is calibrated utility across those conditions, not a single headline accuracy number.

5. Proposed Framework: HITL-CausalFusion

5.1 Design principles

HITL-CausalFusion is guided by six principles:

1. **Evidence before narrative:** every recommendation must link to time-bounded telemetry, topology, or a recorded change.
2. **Topology as a constraint, not ground truth:** observed dependencies restrict implausible edges while allowing uncertain shared-resource and asynchronous relationships.
3. **Reliability-aware multimodality:** missing, delayed, sampled, or low-quality modalities are down-weighted and exposed to the user.
4. **Causal humility:** temporal and structural scores generate hypotheses; interventionally validated effects receive a stronger label.
5. **Calibrated abstention:** the system may state that evidence is insufficient or that multiple causes are indistinguishable.
6. **Human authority and organizational safety:** engineers retain decision authority, and incident data is not repurposed for individual performance scoring.

5.2 Architecture

The framework contains seven layers. First, the ingestion layer collects OpenTelemetry-compatible metrics, logs, traces, deployment and configuration events, topology snapshots, and approved incident knowledge. Second, an alignment layer normalizes identifiers, clocks, windows, and redaction policies while preserving provenance. Third, modality-specific encoders generate service-window representations. Fourth, a reliability gate estimates whether each representation is trustworthy given sampling, missingness, parser confidence, cardinality change, and drift. Fifth, a topology-constrained temporal causal graph connects candidate evidence across services, infrastructure, and changes. Sixth, an uncertainty-aware ranker combines evidence and performs counterfactual consistency checks. Seventh, an explanation interface presents ranked hypotheses, supporting and conflicting evidence, missing signals, confidence, and safe next tests. Human feedback and postmortem outcomes update retrieval indexes, reliability estimates, and evaluation data under governance controls.

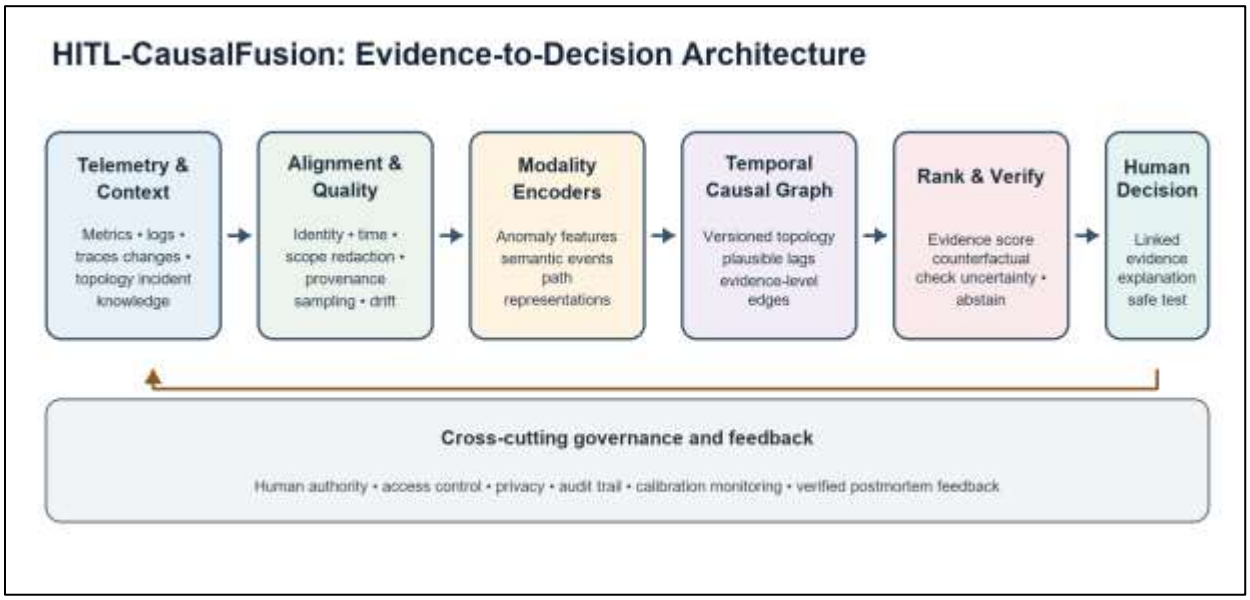

Figure 1. HITL-CausalFusion separates evidence processing, causal hypothesis generation, ranking, explanation, and human authority under cross-cutting governance.

The architecture deliberately separates scoring from explanation. A graph or statistical model produces the candidate set and evidence bundle. An optional LLM converts that structured bundle into incident-appropriate language using constrained generation and citations to internal evidence. This separation makes it possible to test whether the narrative is faithful to the underlying score.

5.3 Representation and fusion

For service or resource s, modality m, and time window t, let X_{s,t}^{(m)} denote raw observations. A modality encoder f_m yields

$$z_{s,t}^{(m)} = f_m(X_{s,t}^{(m)})$$

Metric encoders capture trends, seasonality, and multivariate anomalies; log encoders represent parsed templates, parameters, and semantic events; trace encoders aggregate span attributes while retaining path structure; and change encoders represent deployments, flags, configurations, and ownership. Each encoder emits an anomaly contribution a_{s,t}^{(m)} and a reliability estimate q_{s,t}^{(m)}\in[0,1].

Fusion weights are conditioned on both content and quality:

$$w_{s,t}^{(m)} = q_{s,t}^{(m)} r_{s,t}^{(m)} / \Sigma_{m'} q_{s,t}^{(m')} r_{s,t}^{(m')}$$

where r is a modality-availability indicator that distinguishes absent evidence from normal evidence. The fused state is

$$z_{s,t} = \Sigma_m w_{s,t}^{(m)} z_{s,t}^{(m)}$$

Cross-modal attention may be used after reliability gating to detect evidence such as a deployment followed by a new log template and an abnormal trace path. Gating must remain visible in the explanation so that engineers know, for example, that a candidate depended mainly on logs because trace coverage was 8%.

5.4 Topology-constrained temporal causal graph

Candidate edges are drawn from observed calls, messaging relationships, co-residency, shared dependencies, deployment relations, and operational knowledge. Temporal learning estimates whether changes in a candidate improve prediction of downstream anomalies at plausible lags. Constraints prevent impossible directions—for example, a downstream response cannot cause an already completed upstream request—while allowing feedback across successive requests.

The graph has three edge states: **observed dependency**, **predictive propagation**, and **intervention-supported causal relation**. The distinction is encoded in metadata rather than hidden inside a single edge weight. Hidden-confounder risk is estimated when multiple services share an unobserved or weakly instrumented resource. Topology drift triggers graph re-estimation and lowers confidence until sufficient evidence accumulates.

5.5 Candidate scoring, counterfactual checks, and abstention

For candidate c, the framework computes

$$R(c) = \lambda_1 A(c) + \lambda_2 P(c) + \lambda_3 K(c) + \lambda_4 F(c) - \lambda_5 U(c)$$

where A measures fused anomaly evidence, P measures time-respecting propagation plausibility, K measures prior compatibility with changes and validated incident knowledge, F estimates counterfactual consistency, and U penalizes uncertainty, missingness, drift, and disagreement. The lambdas are learned on a training domain or set by a validation protocol; they must not be tuned on the test incidents.

The counterfactual component asks whether normalizing candidate evidence in a learned structural model reduces predicted downstream symptoms. This is a model-based diagnostic check, not proof of a real-world intervention. Stronger confirmation comes from safe discriminating tests, chaos experiments in nonproduction environments, canary rollback, or naturally occurring recovery. The user interface labels the evidence level.

The system abstains when the maximum calibrated probability is below \tau, the gap between the first and second candidate is below \delta, required modalities are unavailable, the incident is out of distribution, or candidate explanations conflict materially. Abstention returns useful work: unresolved candidates, missing evidence, and the next safe test with the highest expected information gain.

5.6 Explanation and human interaction

Each recommendation contains:

- the candidate and diagnostic granularity;
- a calibrated confidence interval or probability with calibration provenance;
- an event timeline;
- supporting and contradicting evidence;
- the propagation path and its edge types;
- data-quality warnings and missing modalities;
- one or more safe discriminating tests;
- links to approved runbooks or similar incidents; and
- a statement of knowledge limits.

This design reflects NIST's principles of explanation, meaningfulness, explanation accuracy, and knowledge limits [33], as well as the AI RMF characteristics of validity, reliability, transparency, explainability, privacy, and human oversight [34]. GNNExplainer may inform subgraph and feature attribution [35], but post-hoc importance is not automatically a causal explanation. The interface therefore distinguishes "features influencing the model" from "evidence supporting the incident hypothesis."

6. Falsifiable Research Propositions

The framework yields testable propositions rather than assumed benefits.

P1—Reliability-aware fusion. Reliability-gated multimodal fusion will improve macro-averaged top-three localization and expected calibration error relative to early fusion and the best single modality, especially under modality corruption or absence.

P2—Topology-constrained temporal reasoning. Restricting causal discovery with time-versioned topology and plausible lags will reduce symptom-first rankings relative to unconstrained correlation and static random-walk baselines.

P3—Counterfactual consistency. Adding model-based counterfactual checks will improve precision among high-confidence recommendations, though it may lower coverage through increased abstention.

P4—Explanations and appropriate reliance. Evidence-linked explanations with calibrated uncertainty will improve diagnostic correctness and trust calibration compared with rank-only recommendations, without increasing acceptance of deliberately incorrect recommendations.

P5—Human feedback under governance. Structured incident-commander feedback and verified postmortem labels will improve within-organization performance over time, provided labels are separated from employee evaluation and monitored for team and service bias.

These propositions could fail. Late fusion may lose cross-modal interactions; topology constraints may exclude unknown dependencies; counterfactual models may be misspecified; explanations may create unwarranted confidence; and feedback may encode organizational habits rather than technical truth. The evaluation must be designed to expose these failures.

7. Evaluation Protocol

7.1 Evaluation questions and datasets

The protocol evaluates four dimensions: localization, robustness, operational performance, and human use. RCAEval offers 735 failure cases, 11 fault types, three datasets, and metrics, logs, and traces with 15 reproducible baselines [36]. DeathStarBench provides diverse end-to-end microservice applications suitable for controlled deployment and fault injection [3]. OpenRCA adds open-ended enterprise failures and long

multimodal contexts [25]. AIOpsLab supplies an environment for deploying services, injecting faults, executing workloads, exposing telemetry, and scoring agents [24].

No single dataset is sufficient. Benchmark labels make comparisons reproducible but may simplify incident boundaries. Controlled faults establish onset and intervention ground truth but may not represent production novelty. Enterprise cases improve realism but create privacy, access, and label ambiguity. Results should therefore be reported separately by dataset and fault family before any pooled estimate.

7.2 Fault and operating-condition matrix

The experimental matrix should include:

- resource faults: CPU throttling, memory pressure/leak, disk and I/O saturation;
- network faults: latency, loss, DNS failure, partition, connection exhaustion;
- application faults: exceptions, deadlocks, inefficient queries, bad cache behavior;
- dependency faults: database slowdown, queue backlog, third-party timeout;
- change faults: faulty deployment, configuration error, incompatible schema, feature flag;
- cascading and multi-fault incidents; and
- open-set incidents whose fault type was absent from training.

For each fault, vary load, system scale, trace sampling, log loss, metric delay, clock skew, topology churn, and background anomalies. Include clean runs to measure false alarms. Use repeated seeds and preserve the injection manifest, topology version, telemetry, and expected causal chain.

7.3 Baselines and ablations

Baselines should represent distinct paradigms rather than only weak implementations: correlation or anomaly ranking; graph random walk; MicroRCA- or MicroRank-style topology/trace ranking; a causal-discovery pipeline; a multimodal graph model; retrieval-plus-LLM; and an unconstrained agent. Reproducible reference implementations from RCAEval should be preferred where compatible [36].

Ablations remove one design contribution at a time: no reliability gate, no topology constraints, no change events, no counterfactual check, no uncertainty penalty, no abstention, no human feedback, and no evidence grounding for the language layer. Additional leave-one-modality-out experiments quantify dependence on metrics, logs, traces, and changes.

7.4 Metrics

Dimension	Metrics	Interpretation
Localization	Top-1/Top-3 accuracy, mean reciprocal rank, mean average precision, service- and metric-level accuracy	Whether the verified cause appears early enough to guide response
Classification	Macro/micro F1, per-fault recall, open-set AUROC/AUPRC	Whether failure types are recognized without hiding minority classes
Calibration	Expected calibration error, Brier score, reliability plots, risk-coverage curve	Whether confidence corresponds to correctness and abstention is useful
Robustness	Relative degradation under missing modalities, sampling, drift, noise, unseen services/faults	Whether performance survives realistic observability failures
Efficiency	Detection-to-ranking latency, compute/memory, telemetry volume, cost per incident	Whether the method can operate inside incident-response time constraints
Explanation	Evidence precision/recall, faithfulness tests, contradiction rate, human-rated usefulness	Whether the narrative is grounded and discriminating
Operations	Time to first correct hypothesis, time to mitigation, unnecessary diagnostic actions, unsafe recommendation acceptance	Whether assistance improves real work rather than a benchmark label

Top-*k* accuracy must be paired with calibration and coverage. A system that answers 40% of incidents at 90% precision may be safer than one that answers all incidents at 60% precision, depending on incident severity and human workflow. Risk-coverage curves make this trade-off visible.

7.5 Statistical analysis and reproducibility

Use incident-level bootstrap confidence intervals and paired tests because methods evaluate the same cases. For binary correctness, a paired proportion test or hierarchical logistic model is appropriate; for ranking measures and time, use bootstrap or mixed-effects models after inspecting distributions. Report effect sizes and 95% confidence intervals rather than relying on *p*-values alone. Correct planned multiple comparisons and publish the analysis plan before examining final outcomes.

Reproducibility artifacts should include containerized code, model and dependency versions, random seeds, schemas, sampling policies, preprocessing, topology snapshots, injection manifests, prompt templates, model temperatures, token budgets, and hardware. Separate train, validation, and test systems where possible to avoid service-identity leakage. Incident histories retrieved at test time must exclude the evaluated incident and its derivative postmortem.

7.6 Human-subject study

Technical accuracy does not determine operational value. A controlled study should compare three conditions: (A) conventional dashboards and runbooks; (B) an AI-ranked candidate list without explanation; and (C) the full evidence-linked, uncertainty-aware interface. A counterbalanced crossover or Latin-square design can reduce participant and scenario effects.

Recruit approximately 24–36 participants across incident-command, SRE, platform, application, and operations roles and across relevant experience levels. Treat this number as a planning range; determine the final sample through prospective power or precision analysis. Use 6–9 realistic scenarios balanced by fault family and difficulty. Randomly insert some deliberately incorrect high-confidence-looking suggestions to measure automation bias, while ensuring the exercise is conducted in a safe simulation.

Primary outcomes are diagnostic correctness and time to a defensible first hypothesis. Secondary outcomes include mitigation choice, number of unnecessary queries, evidence recall, confidence calibration, recommendation acceptance when the AI is wrong, and recovery after an initial misdirection. NASA-TLX measures mental, physical, temporal, performance, effort, and frustration demands [37]. Trust measures should focus on appropriate reliance rather than general liking; Lee and See define trust as central to reliance decisions and emphasize that automation should support calibrated use [38].

Analyze outcomes with mixed-effects models containing condition and experience as fixed effects and participant and scenario as random effects. Report subgroup estimates carefully and avoid ranking employees. Qualitative interviews should ask which evidence changed the participant's hypothesis, what uncertainty information was understood, and when the participant chose to override the tool.

7.7 Organizational ethics and HR contribution

The human study and operational deployment create a substantive role for an HR or people-operations coauthor. Legitimate contributions include job-role sampling, inclusion criteria, informed-consent language, workload and trust constructs, interview design, organizational-change analysis, psychological-safety safeguards, and interpretation of adoption results. Authorship should reflect actual intellectual and drafting contributions, not job title alone.

Incident telemetry can reveal work patterns, error messages, team ownership, and individual actions. Governance should prohibit reuse for performance appraisal or blame assignment; deidentify participants; minimize and time-limit collection; separate research data from employment records; aggregate reporting; control access by role; and obtain institutional or organizational ethics review where applicable. Participation must be voluntary and nonparticipation must have no employment consequence. These controls align with blameless postmortem practice and reduce incentives to hide errors [26].

8. Analytical Synthesis and Expected Trade-offs

Because the framework has not yet been evaluated, this section describes reasoned expectations and boundary conditions—not results.

First, multimodal fusion is most likely to help when modalities are complementary and correctly aligned. A deployment event may establish a plausible initiating time, logs may identify a new exception, traces may show the first affected path, and metrics may quantify propagation. When all modalities reflect the same downstream symptom, fusion adds confidence without causal information. Reliability gating should therefore improve robustness mainly under heterogeneous quality; on clean, homogeneous benchmarks it may add complexity with little gain.

Second, topology constraints reduce the search space but can encode instrumentation bias. They should improve ranking when runtime maps are complete and versioned, yet harm recall for undocumented dependencies, shared resources, or asynchronous flows. The abstention mechanism is essential when the graph is stale. A useful implementation should reveal which candidate edges were excluded by constraints so that operators can challenge the model.

Third, counterfactual consistency increases precision only to the extent that the structural model is credible. In observational production data, the check is best understood as "would this learned model predict symptom reduction?" Controlled fault injection and canary rollback provide stronger evidence. The framework's three-level edge semantics prevent predictive relations from being presented as interventionally validated causation.

Fourth, LLMs are likely to improve accessibility and evidence synthesis more than core localization. The industrial-scale promise reported by incident-recommendation and RCACopilot studies [22], [23] coexists with the low solve rate in OpenRCA [25]. Structured retrieval and evidence-bounded generation should reduce hallucination, but faithfulness must be tested by removing supporting evidence, swapping candidate labels, and checking whether the narrative changes appropriately.

Fifth, explanations may either calibrate or inflate trust. A polished timeline can make a wrong answer more persuasive. Therefore, human evaluation must include wrong recommendations, conflicting evidence, and abstention. Success means participants accept correct advice efficiently, reject or test incorrect advice, and understand when the system lacks knowledge.

9. Deployment and Governance Blueprint

9.1 Phased adoption

A responsible rollout has four stages. In **offline replay**, the system analyzes historical incidents with hidden labels and no operational effect. In **shadow mode**, it produces recommendations during live incidents but exposes them only to evaluators after decisions are recorded. In **decision support**, on-call staff can view recommendations and explanations while retaining authority. Only after sustained validation should narrowly scoped automated actions be considered, and then only reversible, policy-approved actions with human confirmation.

Promotion gates should include calibration by severity and service, maximum harmful-suggestion rate, minimum abstention behavior under missing telemetry, drift monitoring, red-team tests, explanation faithfulness, privacy review, and documented rollback. Models should be versioned with the topology, telemetry schema, prompt, retrieval index, and policy set used for each recommendation.

9.2 Human roles and accountability

Accountability remains organizational. The incident commander decides operational action; service owners interpret local context; platform teams maintain instrumentation; model teams validate the inference pipeline; security and privacy teams govern data; and people/HR partners safeguard participation, workload assessment, and nonpunitive use. NIST's GOVERN, MAP, MEASURE, and MANAGE functions provide a useful cross-cutting structure [34].

Feedback should be structured as "candidate confirmed," "candidate disproved," "insufficient evidence," "propagation node," or "contributing condition," with a link to the validating action. Free-text feedback can supplement but should not silently become a training label. Disagreement is valuable evidence and should trigger review rather than be scored as operator error.

9.3 Security and privacy

Logs and traces may contain credentials, personal data, customer identifiers, or proprietary business logic. The alignment layer should enforce collection minimization, secret scanning, field-level redaction, retention limits, tenant isolation, access logs, and purpose limitation. Retrieval systems must respect source permissions. Prompt-injection-like content in logs or tickets must be treated as untrusted data, never as executable instruction. Tool calls and remediation require allowlists, parameter validation, simulation where possible, and complete audit trails.

10. Threats to Validity and Limitations

Construct validity. Root-cause labels vary in granularity and may encode the mitigation rather than the initiator. Top-*k* service accuracy can reward finding a victim service. Studies should preserve initiating condition, propagation chain, contributing factors, and repair separately.

Internal validity. Fault injection may create artifacts that reveal the injector. Workload and telemetry differences may leak labels. Hyperparameter tuning on target incidents can inflate performance. Blind evaluation, held-out systems, standardized injectors, and leakage audits are required.

External validity. Public benchmarks cover only a fraction of production stacks and organizational practices. Performance may not transfer across languages, service meshes, observability vendors, or failure cultures. Report domain-specific results and treat transfer as an empirical question.
Causal validity. Temporal precedence, conditional dependence, and graph direction do not guarantee intervention effects. Hidden shared resources, feedback, aggregation, and nonstationarity remain central threats. The proposed framework reduces overclaiming but cannot eliminate them without experiments or credible natural variation.
Human-study validity. Simulated incidents may lack the stress, stakes, interruptions, and team communication of production. Participants may infer the study purpose. Counterbalancing, realistic artifacts, behavioral outcomes, and follow-up field evaluation can reduce—but not remove—these limitations.
LLM limitations. Model updates, nondeterminism, context windows, retrieval errors, and persuasive hallucinations complicate replication and safety. The language layer must be optional, versioned, evidence-grounded, and independently evaluated.
Review limitations. This is a focused integrative review, not a PRISMA-style systematic review or meta-analysis. It does not claim exhaustive coverage or pooled effect estimates. Reported performance values are not directly comparable because datasets, root-cause definitions, and baselines differ.

11. Research Agenda

Five priorities follow from the synthesis. First, benchmarks should include topology and instrumentation drift, concurrent faults, ambiguous labels, missing modalities, and human decision outcomes. Second, causal RCA should publish its identification assumptions and distinguish predictive, counterfactual-model, and intervention-supported claims. Third, multimodal systems should report per-modality reliability and risk-coverage curves, not only aggregate F1. Fourth, LLM agents should be evaluated on evidence faithfulness, safe tool use, and abstention in addition to narrative correctness. Fifth, field research should examine whether assistance reduces cognitive load and restores service faster without weakening expertise, psychological safety, or accountability.
An implementation of HITL-CausalFusion should begin with the smallest falsifiable version: versioned topology, metrics and traces, change events, a reliability gate, a transparent candidate score, and abstention. Logs and an LLM explanation layer can be added after the core ranking is calibrated. This sequence makes it easier to attribute gains and identify failure modes.

12. Conclusion

AI-driven RCA for microservice-based cloud applications has made real progress, but it is not a solved problem. Graph, trace, causal, multimodal, and language-model methods each capture valuable evidence, yet how well they report depends heavily on system boundaries, telemetry quality, diagnostic granularity, and evaluation design. Broad benchmarks make this plain: there is substantial difficulty and no universal winner. Production readiness demands more than top-k accuracy.
HITL-CausalFusion reframes RCA as calibrated, evidence-linked hypothesis generation. It combines reliability-aware multimodal representations, time-versioned topology, temporally constrained causal hypotheses, counterfactual consistency checks, uncertainty penalties, and explicit abstention. Its human interface exposes supporting and conflicting evidence and preserves operator authority. The accompanying evaluation protocol tests localization, calibration, robustness, efficiency, explanation faithfulness, workload, and appropriate reliance. The framework's value remains a hypothesis until those tests are performed; that explicit boundary is a strength, not a limitation. A publishable and operationally responsible RCA system should know not only what it suspects, but why, how uncertain it is, what evidence would disprove it, and when a human must take the lead.

Author Contributions

Vamsi Krishna Bhavana: Conceptualization, Methodology, Investigation, Literature Review, Framework Development, Technical Analysis, Visualization, and Writing—Original Draft.

Rohith Balerao: Conceptualization, Methodology, Literature Review, Human-in-the-Loop and Organizational Governance Analysis, Validation, and Writing—Review & Editing.

Both authors contributed to the interpretation of the literature, development of the proposed HITL-Causal Fusion framework, revision of the manuscript, and approval of the final version.

References

[1] J. Soldani and A. Brogi, "Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey," *ACM Computing Surveys*, vol. 55, no. 3, art. 59, pp. 1–39, 2023, doi: 10.1145/3501297.

[2] P. Zhou, X. Peng, T. Xie, J. Sun, C. Xu, C. Ji, and W. Zhao, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Transactions on Software Engineering*, vol. 47, no. 2, pp. 243–260, 2021, doi: 10.1109/TSE.2018.2887384.

[3] Y. Gan et al., "An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems," in *Proc. ASPLOS*, 2019, pp. 3–18, doi: 10.1145/3293882.3339893.

[4] J. Sillito and T. Kutomi, "Failures and fixes: A study of software system incident response," in *Proc. IEEE ICSME*, 2020, pp. 185–195, doi: 10.1109/ICSME46990.2020.00027.

[5] Q. Wang and Y. Qi, "A comprehensive survey on root cause analysis in (micro) services: Methodologies, challenges, and trends," arXiv:2408.00803, 2024.

[6] B. H. Sigelman et al., "Dapper, a large-scale distributed systems tracing infrastructure," Google Research, 2010. [Online]. Available: https://research.google/pubs/dapper-a-large-scale-distributed-systems-tracing-infrastructure/

[7] OpenTelemetry Authors, "Signals," *OpenTelemetry Documentation*, 2026. [Online]. Available: https://opentelemetry.io/docs/concepts/signals/ (accessed Sep. 3, 2026).

[8] Á. Brandón, M. Solé, A. Huélamo, D. Solans, M. S. Pérez, and V. Muntés-Mulero, "Graph-based root cause analysis for service-oriented and microservice architectures," *Journal of Systems and Software*, vol. 159, art. 110432, 2020, doi: 10.1016/j.jss.2019.110432.

[9] L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "MicroRCA: Root cause localization of performance issues in microservices," in *Proc. IEEE/IFIP NOMS*, 2020, doi: 10.1109/NOMS47738.2020.9110353.

[10] D. Liu et al., "MicroHECL: High-efficient root cause localization in large-scale microservice systems," in *Proc. IEEE/ACM ICSE-SEIP*, 2021, pp. 338–347, doi: 10.1109/ICSE-SEIP52600.2021.00043.

[11] M. Ma, J. Xu, Y. Wang, P. Chen, Z. Zhang, and P. Wang, "AutoMAP: Diagnose your microservice-based web applications automatically," in *Proc. The Web Conference*, 2020, pp. 246–258, doi: 10.1145/3366423.3380111.

[12] Z. Li et al., "Practical root cause localization for microservice systems via trace analysis," in *Proc. IEEE/ACM IWQoS*, 2021, doi: 10.1109/IWQOS52092.2021.9521340.

[13] G. Yu et al., "MicroRank: End-to-end latency issue localization with extended spectrum analysis in microservice environments," in *Proc. The Web Conference*, 2021, pp. 3087–3098, doi: 10.1145/3442381.3449905.

[14] Y. Gan, M. Liang, S. Dev, D. Lo, and C. Delimitrou, "Sage: Practical and scalable ML-driven performance debugging in microservices," in *Proc. ASPLOS*, 2021, pp. 135–151, doi: 10.1145/3445814.3446700.

[15] Y. Zhang et al., "CloudRCA: A root cause analysis framework for cloud computing platforms," in *Proc. ACM CIKM*, 2021, pp. 4373–4382, doi: 10.1145/3459637.3481903.

[16] R. Xin, P. Chen, and Z. Zhao, "CausalRCA: Causal inference based precise fine-grained root cause localization for microservice applications," *Journal of Systems and Software*, vol. 203, art. 111724, 2023, doi: 10.1016/j.jss.2023.111724.

[17] C.-M. Lin, C. Chang, W.-Y. Wang, K.-D. Wang, and W.-C. Peng, "Root cause analysis in microservice using neural Granger causal discovery," in *Proc. AAAI*, vol. 38, no. 1, 2024, pp. 206–213, doi: 10.1609/aaai.v38i1.27772.

[18] C. Lee, T. Yang, Z. Chen, Y. Su, and M. R. Lyu, "Eadro: An end-to-end troubleshooting framework for microservices on multi-source data," in *Proc. IEEE/ACM ICSE*, 2023, doi: 10.1109/ICSE48619.2023.00150.

[19] S. Zhang et al., "Robust failure diagnosis of microservice system through multimodal data," *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 3851–3864, 2023, doi: 10.1109/TSC.2023.3290018.

[20] C. Duan et al., "Famos: Fault diagnosis for microservice systems through effective multi-modal data fusion," in *Proc. IEEE/ACM ICSE*, 2025, pp. 2613–2624, doi: 10.1109/ICSE55347.2025.00073.

[21] Y. Han, Q. Du, Y. Huang, P. Li, X. Shi, J. Wu, P. Fang, F. Tian, and C. He, "Holistic root cause analysis for failures in cloud-native systems through observability data," *IEEE Transactions on Services Computing*, vol. 17, no. 6, pp. 3789–3802, 2024, doi: 10.1109/TSC.2024.3478759.

[22] T. Ahmed et al., "Recommending root-cause and mitigation steps for cloud incidents using large language models," in *Proc. IEEE/ACM ICSE*, 2023, pp. 1737–1749, doi: 10.1109/ICSE48619.2023.00149.
[23] Y. Chen et al., "Automatic root cause analysis via large language models for cloud incidents," in *Proc. ACM EuroSys*, 2024, pp. 674–688, doi: 10.1145/3627703.3629553.
[24] Y. Chen et al., "AIOpsLab: A holistic framework to evaluate AI agents for enabling autonomous clouds," *Proceedings of Machine Learning and Systems*, vol. 7, 2025.
[25] J. Xu et al., "OpenRCA: Can large language models locate the root cause of software failures?" in *Proc. ICLR*, 2025. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2025/hash/d29b8d53678015079e1d245c023e49d2-Abstract-Conference.html
[26] D. Rogers et al., "Postmortem culture: Learning from failure," in *The Site Reliability Workbook*. Google, 2018. [Online]. Available: https://sre.google/workbook/postmortem-culture/
[27] C. Chan and B. Cooper, "Debugging incidents in Google's distributed systems," *Communications of the ACM*, vol. 63, no. 10, pp. 40–46, 2020, doi: 10.1145/3397880.
[28] R. Balerao, "Agentic AI framework for autonomous workforce analytics and decision support in enterprise HRIS systems," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 18, no. 5s, pp. 611–621, 2026, doi: 10.70917/ijcisim-2026-2756.
[29] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *Proc. IEEE ICWS*, 2017, pp. 33–40, doi: 10.1109/ICWS.2017.13.
[30] Y. Su et al., "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. ACM KDD*, 2019, pp. 2828–2837, doi: 10.1145/3292500.3330672.
[31] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "DAGs with NO TEARS: Continuous optimization for structure learning," in *Advances in Neural Information Processing Systems 31*, 2018.
[32] L. Pham, H. Ha, and H. Zhang, "Root cause analysis for microservice system based on causal inference: How far are we?" in *Proc. IEEE/ACM ASE*, 2024, doi: 10.1145/3691620.3695065.
[33] P. J. Phillips et al., *Four Principles of Explainable Artificial Intelligence*, NISTIR 8312. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2021, doi: 10.6028/NIST.IR.8312.
[34] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, 2023, doi: 10.6028/NIST.AI.100-1.
[35] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 9240–9251.
[36] L. Pham, H. Zhang, H. Ha, F. D. Salim, and X. Zhang, "RCAEval: A benchmark for root cause analysis of microservice systems with telemetry data," in *Companion Proc. The Web Conference*, 2025, doi: 10.1145/3701716.3715290.
[37] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research," in *Human Mental Workload*, P. A. Hancock and N. Meshkati, Eds. Amsterdam, The Netherlands: North-Holland, 1988, pp. 139–183, doi: 10.1016/S0166-4115(08)62386-9.
[38] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance," *Human Factors*, vol. 46, no. 1, pp. 50–80, 2004, doi: 10.1518/hfes.46.1.50_30392.