SvedbergOpen
DISSEMINATIONOFKNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

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

Research Paper Open Access

From Alert Fatigue to Autonomous Diagnosis: An Agentic AI Framework for Root-Cause Analysis in Cloud Operations

Sai Nitesh Palamakula

Independent Researcher, India. Email: gowthampappulaconnects@gmail.com

Abstract
Incident volume in large cloud estates has outpaced the capacity of manual root-cause analysis, and the diagnostic step, rather than detection, is where operator time now accumulates. Recent industrial work has applied large language models to this problem, including deployed retrieval-augmented pipelines that match an incident against historical cases and generate an explanation. This paper does not claim that territory. It proposes a narrower architecture built on two observations those systems leave unaddressed. First, the recorded failure modes of multiagent language model systems are failures of specification and of inter-agent verification rather than of any single agent's reasoning, which argues for decomposing diagnosis into roles whose outputs are checked by a different role before being acted on. Second, an accuracy figure cannot distinguish a correct diagnosis from a confident wrong one, and existing cloud root-cause systems report accuracy without reporting calibration, leaving the operator no basis for deciding how far to rely on a given hypothesis. The paper specifies four agent roles with an explicit verification point at each handoff, a calibrated confidence signal surfaced to the operator as a reliance cue rather than as a score, and an evaluation design fixed in advance, including a falsifiable proposition that the relationship between granted autonomy and net benefit is concave and that its turning point moves with retrieval grounding quality. Nothing described here has been built, deployed or measured. The contribution is the architecture and the evaluation design, and the paper states throughout which claims would require which measurements to support.

Keywords: root-cause analysis, cloud incident management, AIOps, large language model agents, multi-agent systems, retrieval-augmented generation, confidence calibration, trust in automation, alert fatigue, site reliability engineering, design science, human-in-the-loop

I. Introduction

The bottleneck in incident response has moved. Detection is substantially automated; an empirical account of the incident lifecycle at a large cloud provider shows that the interval between detection and diagnosis is where on-call time accumulates [2]. A survey of AIOps methods for failure management reaches the same conclusion structurally, identifying root-cause analysis as the least mature stage of the pipeline [1].

The reason is not that the problem is unstudied. Root-cause analysis in microservice systems has a substantial literature, taxonomized in a recent survey [4], with dependency-graph approaches that localize a faulty service from its position in the call graph [5] and trace-based approaches that localize from anomalous request traces [6]. What that work shares is an assumption that the output is a localization: a service, a component, a metric. An operator does not need a localization. They need to know what happened and whether the answer is trustworthy enough to act on at three in the morning.

What has already been done

Recent industrial work addresses precisely that gap. Large language models have been evaluated on real production incidents, generating root cause and mitigation recommendations [14]. A retrieval-augmented pipeline has been deployed that collects diagnostic context, matches the incident against historical cases and generates an explanation [15]. Agents in the reason-then-act style [11], equipped with retrieval tools, have been evaluated on root-cause analysis against simpler baselines [16].

This paper is not proposing to do that. It is proposing something narrower, and the narrowness is the point.

What remains

Two problems are visible in that body of work and are not addressed by it.

The architecture is undecomposed
[14] applies a model. [15] applies a pipeline. [16] applies an agent with tools. None separates diagnosis into roles whose conclusions are checked by a different role before the next step proceeds. That matters because an empirically derived taxonomy of multi-agent language model failures finds the dominant failure categories to be specification problems and inter-agent misalignment rather than failures of individual reasoning [13]. A system built without explicit verification between stages is built against no model of how such systems actually fail.

The confidence problem is unaddressed
Retrieval reduces ungrounded generation but does not eliminate it; hallucination persists even when output is conditioned on retrieved passages, as a dedicated corpus demonstrates [10]. Modern neural networks are systematically overconfident by default [19]. Trust in automation research establishes that reliance must be calibrated to actual reliability, and that miscalibrated reliance is the failure mode that matters [17]. Yet no cloud root-cause system in the set above reports a calibrated confidence or measures operator reliance. Reporting accuracy alone tells an operator how often the system is right in aggregate. It does not tell them whether this answer, now, is one of the right ones.

Contributions
Stated with their evidence status, since none has been evaluated:

A role-decomposed agent architecture with explicit verification at each handoff (Section IV), designed against the failure taxonomy in [13]. Proposed, not built.

A confidence signal specified as a reliance cue (Section V), grounded in [17] to [19]. Proposed, not evaluated.

An evaluation design fixed in advance (Section VI), following design science guidance that an artifact must be rigorously evaluated [20], including the measurements that would refute the architecture.
A falsifiable proposition about autonomy (Section VII): that net benefit is concave in granted autonomy and that the turning point moves outward as retrieval grounding improves.
Section II characterizes the problem. Section III positions the work against what exists. Sections IV and V give the architecture. Sections VI and VII give the evaluation design and the proposition. Sections VIII and IX state the limitations and conclude.
The problem

Volume. Alert storms in production systems have been characterized empirically, and the response in that literature is correlation and suppression, which reduces the count an operator sees without changing what they must do with what remains [3].

Where the time goes. The industrial lifecycle study is the more useful evidence: it locates the cost in the interval between an alert being believed and a cause being established [2]. That is a reasoning interval, not a filtering one.

Why the reasoning is hard. Diagnosis in a distributed system is a multi-step inference across heterogeneous evidence. It requires reading telemetry for a window whose boundaries are themselves a judgment, placing the affected service in a dependency graph that changes with every deployment, recalling whether anything similar has happened before, and producing a cause that the available evidence actually supports. Each step can be done well and the conclusion still be wrong, because an error in the first step propagates silently through the rest.

The human consequence, stated carefully. The construct of alert fatigue describes desensitization arising from exposure to alarms of low predictive value, and it has measured consequences in clinical settings [7]. The mechanism is plausible in operations and the conditions documented in [3] are consistent with it. It has not been measured there, and this paper does not assert that it has. What can be said without overreach is narrower and sufficient: an operator who cannot tell which diagnoses to trust must verify all of them, which returns the cost the automation was meant to remove.
Position against existing work

Table I states what each closest work established and what this paper claims beyond it. The table is written to be read adversarially, because the honest position is that most of the obvious territory is taken.

Prior work	What it established	What it did not address
Ahmed et al. [14]	Language models generating root cause and mitigation recommendations, evaluated on real production incidents	A single model applied end to end. No role separation, no inter-agent verification, no calibration of the confidence attached to a recommendation
Chen et al. [15]	A deployed retrieval-augmented pipeline: collect diagnostic context, match historical incidents, generate an explanation	The pipeline stages are sequential rather than mutually checking. Confidence is not reported as a calibrated quantity the operator can act on
Roy et al. [16]	Reason-then-act agents with retrieval tools evaluated on root-cause analysis against simpler baselines	A single agent with tools rather than separable roles with verification. Reports task performance, not operator reliance
Soldani and Brogi [4], Wu et al. [5], Li et al. [6]	Root-cause localization in microservices from dependency graphs and traces	Output is a localization rather than an explanation, and the evaluation question is localization accuracy rather than whether an operator can act on it

TABLE I. What is already done, and what is left.

The claim this paper can defend, stated plainly: not that a language model with retrieval can diagnose a cloud incident, which is published, deployed and measured, but that decomposing diagnosis into roles that check one another addresses a documented class of multi-agent failure [13] that the published systems were not designed against, and that a calibrated confidence signal addresses a reliance problem [17] that none of them reports on.

The claim this paper cannot defend is that either change improves anything, because neither has been built.



Figure 1. Where published work already reaches, and what is left. The retrieval-augmented half of the proposed architecture is published and industrially evaluated. The contribution claimed here is narrower.

The architecture

Four roles, each with a defined input, a defined output, and a verification point at the handoff.

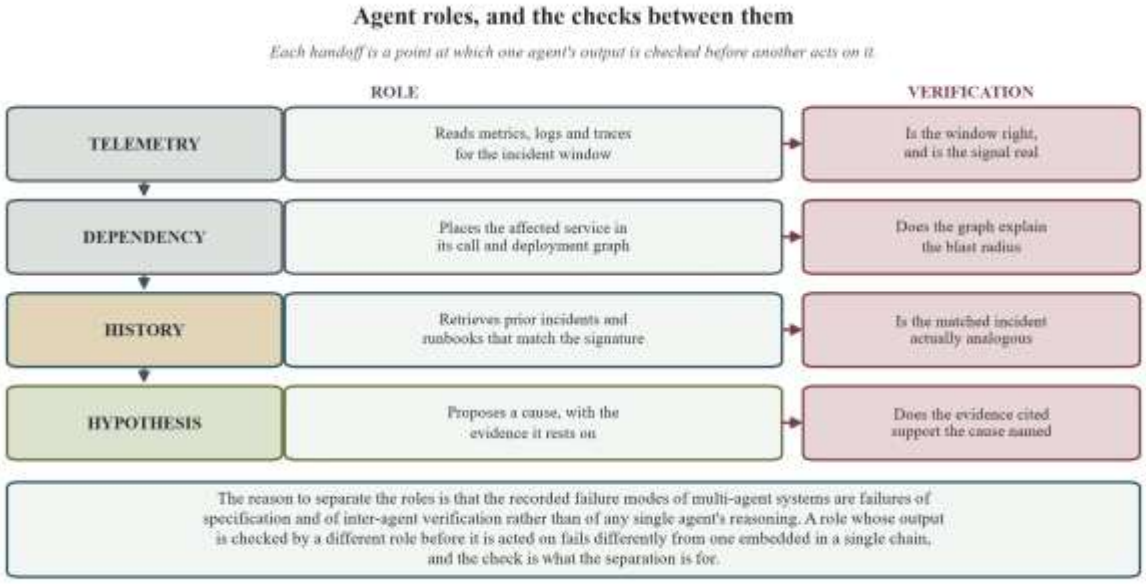


Figure 2. The agent roles and the check at each handoff. Each verification asks a question about the previous role's output that the previous role is not well placed to ask about itself.

Telemetry. Reads metrics, logs and traces for the incident window. The verification at its output asks whether the window is right and whether the signal is real, which is a question about the framing of the evidence rather than about its content.

Dependency. Places the affected service in its call and deployment graph, the approach established in the localization literature [5], [6]. The verification asks whether the graph explains the observed blast radius; a proposed locus that cannot account for what else broke is probably wrong.

History. Retrieves prior incidents and runbooks matching the signature, which is the mechanism [15] deploys. The verification asks whether the matched incident is actually analogous rather than merely similar in surface features, which is where retrieval most often misleads. The design space for this role, covering how retrieval, augmentation and generation interact and where each stage degrades, is surveyed in [9]; the point relevant here is that the retrieval stage is a separate source of error from the generation stage, which is an argument for checking it separately.

Hypothesis. Proposes a cause together with the evidence it rests on. The verification asks whether the cited evidence supports the cause named, which is the check that retrieval-augmented generation most needs, since grounding reduces but does not remove ungrounded assertion [10].

Why separation rather than one capable agent. The argument is not that four models outperform one, and the paper does not claim it. It is that the recorded failures of multi-agent systems concentrate in specification and interagent verification [13], so a design in which each handoff carries an explicit check is a design built against the known failure surface rather than around it. Whether that reduces failures is an empirical question, and Section VI states how it would be answered.

What this inherits. Every role runs on a language model, so every role inherits the documented tendency of such systems to produce fluent, ungrounded output [10], and the survey literature on language model agents catalogues the architectural choices and their evaluation problems [12]. Separation does not repair that. It localizes it: a role that fails is a role whose output is checked, rather than a step buried in a chain whose conclusion arrives whole.

Confidence as a reliance signal

The problem with accuracy. A system reporting that it is right most of the time tells an operator nothing about the case in front of them. Table II sets out the four states.

Confidence	Diagnosis	Outcome
High	Correct	Acted on. The intended case
Low	Correct	Verified by hand. Slower, but safe
Low	Wrong	Rejected. The signal did its job
High	Wrong	The failure that matters

TABLE II. Confidence and correctness. Only the fourth row is dangerous, and an aggregate accuracy figure cannot distinguish it from the first.

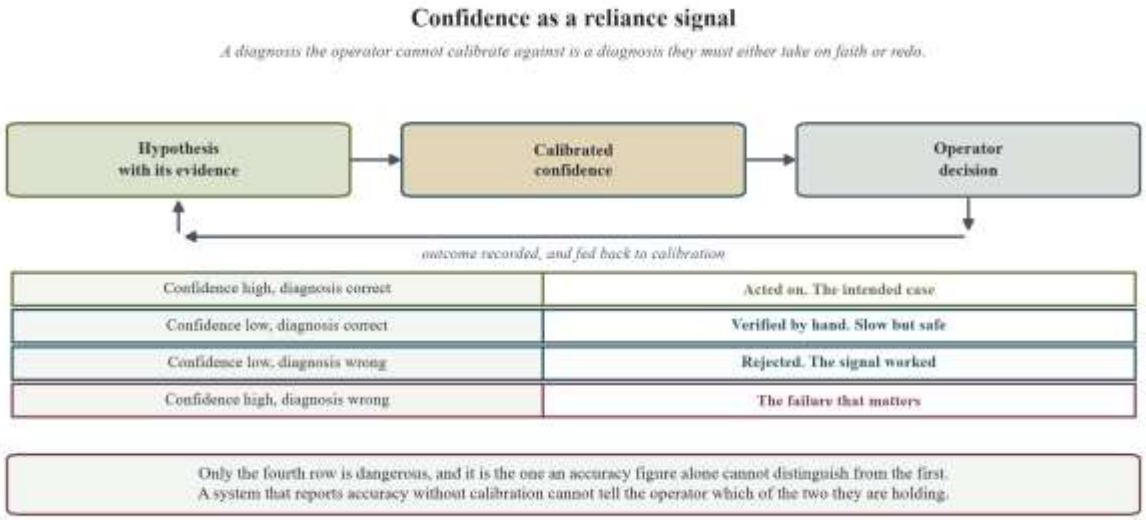


Figure 3. The confidence and reliance loop. Outcomes are recorded and fed back to calibration, so the signal is answerable to its own track record.

Why this is not solved by default. Neural networks are systematically overconfident, and post-hoc correction is required to make a reported confidence correspond to an observed frequency [19]. A system that emits a confidence without calibrating it emits a number that looks like evidence and is not.

What the signal must do. Trust research distinguishes trust from appropriate reliance: the goal is not that operators trust the system more, but that reliance tracks reliability [17]. The levels-of-automation literature makes the same point structurally, that the question is which stage of the task is automated and with what authority, not how much automation there is [18]. A confidence signal is therefore a design element of the human-automation interface rather than a model output, and it should be evaluated as such.

Specification

Confidence is reported per hypothesis, calibrated against recorded outcomes, and presented with the evidence the hypothesis rests on rather than as a bare number. The operator's decision and the eventual outcome are recorded and returned to calibration, which is what makes the signal answerable to its own history rather than to a fixed prior.

Evaluation design

No results are reported. This section specifies what would have to be measured, in the order it should be measured, so that the claims in Sections IV and V are falsifiable in principle. It follows design science guidance that a proposed artifact must be evaluated against criteria set in advance [20].

Quantity	Method	What it would establish	What would refute the design
Verification catch rate per handoff	Replay a labelled incident set; record how often each check rejects an	Whether the separation does work, per role	Checks that almost never fire, meaning the separation costs latency and buys nothing

	upstream output that later proves wrong		
End-to-end diagnostic accuracy against a single-agent baseline	Same incident set through the decomposed architecture and through a single agent with the same tools	Whether decomposition helps at all	No difference, or the baseline performing better
Calibration error of the confidence signal	Compare reported confidence against observed correctness frequency across bins	Whether confidence means anything	Reported confidence uncorrelated with correctness
Operator reliance behaviour	Controlled study; measure acceptance and override against the system's actual correctness	Whether the signal changes reliance in the right direction	Reliance insensitive to confidence, or high-confidence wrong answers accepted at the same rate as highconfidence right ones
Time to root cause against current practice	Paired comparison on matched incidents	The delivery claim	No improvement, or improvement that disappears once verification time is counted
Operational overhead	Instrument inference cost and added latency per incident	Feasibility	Overhead exceeding the time saved

TABLE III. The measurement programme this paper does not have.

The order matters. Calibration should be measured before reliance, because an uncalibrated signal makes the reliance study uninterpretable. Verification catch rate should be measured before end-to-end accuracy, because if the checks never fire the architecture has no mechanism and any accuracy difference is attributable to something else.

The evaluation this paper most needs, and the one most often skipped, is the fourth row. The field optimizes diagnostic accuracy. Whether an operator can distinguish a correct diagnosis from a confident wrong one is a different question, it is the question this architecture exists to address, and it requires human subjects rather than replay.

The autonomy proposition

The framework proposes a specific shape for the relationship between autonomy and benefit, and states it as a proposition rather than a finding because it has not been tested.

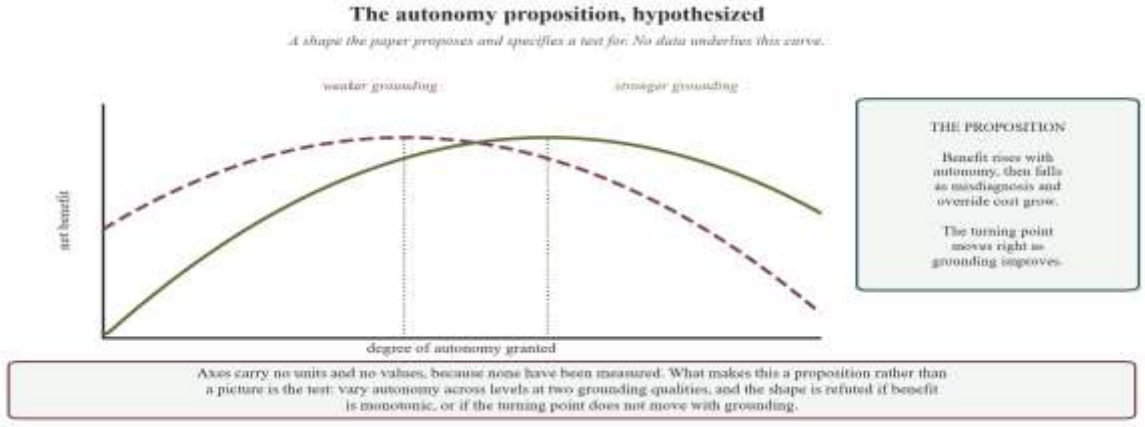


Figure 4. The proposition, hypothesized. The axes carry no units and no values because none have been measured.

The proposition. Net benefit rises with the degree of autonomy granted to the diagnostic system, reaches a maximum, and then declines as the cost of misdiagnosis and of operator override grows faster than the time saved. The turning point moves outward as retrieval grounding quality improves, because better grounding reduces the rate at which additional autonomy produces wrong answers.

Why it is plausible rather than arbitrary. The declining limb rests on two documented effects rather than on intuition. Ungrounded output persists under retrieval [10], so the error rate does not fall to zero with more context. And overconfidence is the default [19], so errors arrive wearing the same confidence as correct answers, which is what makes them expensive to filter.

What would refute it. Three things, and they are stated because a proposition that cannot fail is decoration. If net benefit is monotonic in autonomy across the tested range, the shape is wrong. If the turning point does not move with grounding quality, the moderation claim is wrong. If the predicted interval on the turning point is wide enough to span the tested levels, the experiment cannot discriminate and the proposition is untestable as posed, which is a result about the design rather than about the world.

How it would be tested. Vary granted autonomy across discrete levels, corresponding to the stages at which the system may act without confirmation, at two grounding qualities, and measure net benefit as a composite of time to root cause and the cost of incorrect action. The levels-of-automation framework supplies the vocabulary for defining the autonomy levels precisely enough to vary them [18].

Limitations

No artifact. The architecture has not been implemented. Every claim about its behaviour is a design argument.

No evaluation. No accuracy, latency, calibration or reliance measurement exists, and Section VI is a plan rather than a report.

The closest prior work is evaluated and this is not. [14], [15] and [16] carry industrial evaluation. A reader comparing this paper against them should weight that difference heavily, and Section III states it rather than leaving it to inference.

The alert fatigue construct is borrowed. Its measured evidence base is clinical [7]. Its application to cloud operations is argued from the conditions in [3] and is not itself measured.

Two canonical sources are preprints. The retrieval-augmented generation formulation [8] and the reason-then-act pattern [11] are cited from their preprint records, which is how they are indexed, along with the calibration work [19]. Each is the canonical source for its idea.

The failure taxonomy is recent and singular. The design rests substantially on [13] for its account of how multiagent systems fail. It is one taxonomy, recently published, and the architecture inherits whatever its limitations turn out to be.

No cost model. Four roles rather than one implies more inference per incident. The overhead row in Table III exists because that cost could exceed the benefit, and nothing here establishes that it does not.

Conclusion

The diagnostic step is where incident response now spends its time, and applying language models to it is no longer novel: it is published, deployed and measured. A paper proposing to do it again would be proposing nothing.

What has not been addressed is narrower and, the paper argues, load-bearing. Multi-agent systems fail predominantly through specification and inter-agent verification rather than through the reasoning of any single agent, so a diagnostic architecture ought to be decomposed into roles that check one another at each handoff, which the deployed systems are not. And an operator deciding whether to act on a diagnosis needs a confidence they can calibrate against, which none of those systems reports, because an aggregate accuracy figure cannot distinguish a correct diagnosis from a confident wrong one.

Neither of those is demonstrated here. The architecture is specified, the evaluation is specified in advance and in order, and the autonomy proposition is stated in a form that can fail. What would turn this into a contribution rather than a proposal is the fourth row of Table III: whether an operator, given this signal, relies on the system in proportion to how often it is right.

References

1. P. Notaro, J. Cardoso, and M. Gerndt, "A survey of AIOps methods for failure management," ACM Transactions on Intelligent Systems and Technology, vol. 12, no. 6, pp. 1-45, 2021. doi: 10.1145/3483424
2. S. Ghosh, M. Shetty, C. Bansal, and S. Nath, "How to fight production incidents? An empirical study on a large-scale cloud service," in Proc. 13th Symposium on Cloud Computing (SoCC), 2022, pp. 126-141. doi: 10.1145/3542929.3563482
3. N. Zhao, F. Wang, Z. Li, X. Nie, H. Zhang, Y. Sui, et al., "Understanding and handling alert storm for online service systems," in Proc. IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2020, pp. 162-171. doi: 10.1145/3377813.3381363
4. 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, pp. 1-39, 2022. doi: 10.1145/3501297
5. L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "MicroRCA: root cause localization of performance issues in microservices," in Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS), 2020, pp. 1-9. doi: 10.1109/NOMS47738.2020.9110353
6. S. Sendelbach and M. Funk, "Alarm fatigue: a patient safety concern," AACN Advanced Critical Care, vol. 24, no. 4, pp. 378-386, 2013. doi: 10.1097/NCI.0b013e3182a903f9
7. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," arXiv preprint arXiv:2005.11401, 2020. doi: 10.48550/arXiv.2005.11401
8. Y. Huang and J. X. Huang, "A survey on retrieval-augmented text generation for large language models," ACM Computing Surveys, vol. 58, no. 12, pp. 1-38, 2026. doi: 10.1145/3805774
9. C. Niu, Y. Wu, J. Zhu, S. Xu, K. Shum, R. Zhong, et al., "RAGTruth: a hallucination corpus for developing trustworthy retrievalaugmented language models," in Proc. 62nd Annual Meeting of the Association for Computational Linguistics, 2024, pp. 1086210878. doi: 10.18653/v1/2024.acl-long.585
10. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: synergizing reasoning and acting in language models," arXiv preprint arXiv:2210.03629, 2022. doi: 10.48550/arXiv.2210.03629
11. L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, et al., "A survey on large language model based autonomous agents," Frontiers of Computer Science, vol. 18, no. 6, art. no. 186345, 2024. doi: 10.1007/s11704-024-40231-1
12. M. Cemri, M. Z. Pan, S. Yang, L. A. Agrawal, B. Chopra, R. Tiwari, et al., "Why do multi-agent LLM systems fail?," in Advances in Neural Information Processing Systems, 2025, pp. 135676-135729. doi: 10.52202/085713-4082
13. T. Ahmed, S. Ghosh, C. Bansal, and T. Zimmermann, "Recommending root-cause and mitigation steps for cloud incidents using large language models," in Proc. IEEE/ACM 45th International Conference on Software Engineering (ICSE), 2023, pp. 17371749. doi: 10.1109/ICSE48619.2023.00149
14. Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, et al., "Automatic root cause analysis via large language models for cloud incidents," in Proc. 19th European Conference on Computer Systems (EuroSys), 2024, pp. 674-688. doi: 10.1145/3627703.3629553
15. D. Roy, X. Zhang, R. Bhave, and C. Bansal, "Exploring LLM-based agents for root cause analysis," in Companion Proc. 32nd ACM International Conference on the Foundations of Software Engineering, 2024, pp. 208-219. doi: 10.1145/3663529.3663841
16. J. D. Lee and K. A. See, "Trust in automation: designing for appropriate reliance," Human Factors, vol. 46, no. 1, pp. 50-80, 2004.
17. doi: 10.1518/hfes.46.1.50_30392
18. R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, vol. 30, no. 3, pp. 286-297, 2000. doi: 10.1109/3468.844354
19. C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," arXiv preprint arXiv:1706.04599, 2017. doi: 10.48550/arXiv.1706.04599
20. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," MIS Quarterly, vol. 28, no. 1, pp. 75-106, 2004. doi: 10.2307/25148625