

Designing Multi Agent Workflows for Root Cause Analysis in Distributed Systems

Sneha Gullapalli

Principal Software Engineer | Distributed Systems and Observability

Abstract

Root cause analysis in a distributed system rarely fails because telemetry is entirely absent. More often, the evidence is fragmented across metrics, logs, traces, deployment records, and service ownership boundaries. A single general-purpose agent can search these sources, but it tends to mix observation with interpretation and may settle on the first plausible explanation. This paper proposes a multi-agent workflow that separates evidence collection, hypothesis formation, adversarial verification, and incident reporting. Specialized agents operate over a shared incident record and must attach provenance to every material claim. A coordinator controls scope and budget but cannot declare a root cause by itself. A verifier tests competing hypotheses against topology, time, and counterevidence before a human incident commander accepts the diagnosis. The design treats the language model as a reasoning aid rather than an authority. It is intended as a reference architecture; no production deployment or comparative experiment is claimed.

Keywords root cause analysis, multi-agent systems, distributed systems, observability, telemetry, incident response

1 Problem Definition

A customer-visible timeout may first appear at an API gateway even when the initiating fault sits several hops away: a saturated storage node, a delayed metadata lookup, a failed deployment, or an overloaded database shard. Metrics show the change in system state, traces expose the request path, and logs record local events. None of these signals proves causation on its own. OpenTelemetry similarly treats traces, metrics, and logs as distinct signals that become more useful when they share context and consistent resource attributes [1, 2].

Manual investigation becomes slow when an engineer must pivot between tools, reconstruct topology, compare timestamps, and ask several service teams for context. A general-purpose AI assistant reduces search effort, but an unconstrained assistant introduces a different risk. It can mistake correlation for cause, overlook missing telemetry, or produce a fluent explanation that has no reproducible evidence trail. The design goal is therefore narrower than fully autonomous incident response: produce a ranked, reviewable diagnosis whose supporting observations can be inspected by an engineer.

2 Architecture

The proposed workflow uses an orchestrator and five bounded agents. Each agent receives only the tools and data needed for its role. This division follows established orchestrator-worker and evaluator patterns, but applies them to incident evidence rather than open-ended content generation [3].

Agent	Primary responsibility	Required output
Coordinator	Defines the incident window, scope, and investigation budget	Task plan and completion criteria
Topology	Builds the service and infrastructure dependency graph	Versioned nodes, edges, and ownership
Telemetry	Finds anomalies across metrics, logs, and traces	Evidence records with query, time, and source

Agent	Primary responsibility	Required output
Change	Reviews deployments, configuration, and operational events	Chronology of changes near the incident
Hypothesis	Generates and ranks possible failure mechanisms	Testable hypotheses and expected observations
Verifier	Searches for contradictions and tests causal consistency	Confidence assessment and unresolved gaps

Table 1 Agent responsibilities and evidence contracts

3 Shared Incident Record

Agents should communicate through structured records, not free-form conversation alone. An evidence record contains an immutable identifier, source system, exact query or retrieval action, observation window, component identity, returned value, and collection time. A hypothesis record names a suspected component and failure mechanism, lists evidence for and against it, states what additional observation would disprove it, and records a confidence level. This schema lets the verifier trace a conclusion back to the original telemetry and lets engineers replay the investigation after the incident.

Correlation depends on identity. Trace and span identifiers connect request activity; resource attributes identify services, clusters, nodes, and versions; deployment identifiers link behavior changes to releases. OpenTelemetry context propagation provides a standard mechanism for carrying execution-scoped values across process boundaries [2]. When context is incomplete, the workflow must label the correlation as temporal or inferential rather than exact.

4 Investigation Workflow

4.1 Establish the incident boundary

The coordinator starts with the affected service-level indicator, the first known symptom, the time window, and the suspected blast radius. It then asks the topology agent for the dependency subgraph that could influence the symptom. Constraining the graph prevents every agent from searching the entire environment and reduces both query cost and irrelevant evidence.

4.2 Collect evidence in parallel

The telemetry and change agents work independently. The telemetry agent compares the incident interval with an appropriate baseline and searches for changes in latency distributions, error rates, queue depth, resource saturation, and trace structure. The change agent checks deployments, configuration revisions, scaling events, certificate changes, and maintenance activity. Parallel collection reduces anchoring because one agent does not see the other agent’s preferred explanation before recording its observations.

4.3 Generate testable hypotheses

The hypothesis agent converts observations into failure mechanisms. For example, a sharp increase in object-read latency on one node is only a symptom. A useful hypothesis might be that disk contention on that node delayed a storage operation, which increased downstream request latency. The hypothesis must predict additional evidence, such as elevated device wait time, longer storage spans, and concentration of affected requests on the same node. If those predictions are absent, confidence should fall.

4.4 Verify before reporting

The verifier performs three checks. Temporal consistency asks whether the proposed cause preceded the symptom. Topological consistency asks whether a plausible dependency path connects cause and impact. Counterfactual consistency asks whether unaffected components or requests contradict the explanation. The verifier should also compare at least one alternative hypothesis. Microsoft’s production research on incident triage shows why multiple

roles and decentralized knowledge matter when incident data is heterogeneous and responsibility spans teams [4]. The verifier's purpose, however, is not voting by majority. It is to expose unsupported steps in the causal chain.

5 Confidence and Human Control

A confidence score should be derived from explicit factors rather than the model's self-reported certainty. A practical score can combine evidence coverage, source reliability, temporal fit, topology fit, reproduction against historical telemetry, and the strength of counterevidence. The report should show the factor values so an incident commander can understand why one hypothesis ranks above another.

The workflow should remain read-only during diagnosis. Restarting services, changing replication, moving data, or modifying traffic requires a separate remediation policy and human approval. NIST guidance calls for documented roles, defined human oversight, and evaluation of oversight procedures for higher-risk AI uses [5]. In infrastructure operations, the approval record should bind the proposed action to a specific target, command, time window, rollback plan, and expected verification signal.

6 Failure Modes and Controls

The first failure mode is telemetry ambiguity. Sampling, clock skew, missing trace context, and inconsistent service names can make a correct cause indistinguishable from a nearby symptom. The system should report insufficient evidence rather than fill the gap with a narrative. The second is correlated-agent error: several agents may inherit the same incorrect topology or stale runbook. Independent retrieval and source timestamps reduce this risk. The third is investigation explosion. A coordinator needs limits on graph depth, query volume, model calls, and elapsed time, with escalation when the budget expires. The fourth is prompt injection through logs or tickets. Telemetry must be treated as untrusted data, and tool permissions must not be derived from text found inside an incident record.

Operational evaluation should use replayable incidents with known outcomes. Useful measures include top-one and top-three localization accuracy, time to a defensible hypothesis, evidence precision, unsupported-claim rate, query cost, and engineer acceptance. Recurring incidents are a sensible first deployment target because runbooks and historical outcomes provide stronger evaluation criteria; prior work on workflow-based diagnosis has also evaluated this narrower setting [6].

7 Conclusion

A multi-agent RCA system is valuable when specialization creates checks that a single assistant cannot reliably perform. The proposed design separates collection from interpretation, requires provenance for every conclusion, tests alternatives, and keeps remediation under human control. Its output is not a verdict. It is an evidence package that shortens an engineer's path from symptom to a defensible root-cause hypothesis. Production adoption should begin with read-only analysis of recurring incidents, measured against replayable cases, before expanding to unfamiliar failures or automated corrective action.

Declaration of AI Assisted Writing

OpenAI Codex [7] was used during preparation of this manuscript to assist with development of the initial draft, organization of the technical discussion, language editing, reference discovery, and document formatting. The author supplied the subject matter, professional context, technical direction, and review criteria. The author subsequently reviewed and revised the manuscript, verified the cited sources and technical statements, and accepts full responsibility for the accuracy, originality, interpretation, and final content of the work.

References

- [1] OpenTelemetry Authors, "Signals," OpenTelemetry Documentation. [Online]. Available: <https://opentelemetry.io/docs/concepts/signals/>.
- [2] OpenTelemetry Authors, "OpenTelemetry Specification Overview," OpenTelemetry Documentation. [Online]. Available: <https://opentelemetry.io/docs/specs/otel/overview/>.
- [3] E. Schluntz and B. Zhang, "Building Effective Agents," Anthropic, Dec. 19, 2024. [Online]. Available: <https://www.anthropic.com/research/building-effective-agents>.
- [4] Z. Yu, A. Fang, M. Ma, J. S. Walia, C. Zhang, S. Chi, Z. Li, M. Chintalapati, X. Zhang, R. Wang, C. Bansal, S. Rajmohan, Q. Lin, S. Zhang, D. Pei, and P. He, "Triangle: Empowering Incident Triage with Multi-Agent," in Proc. 40th IEEE/ACM Int. Conf. Automated Software Engineering (ASE), Seoul, Republic of Korea, 2025, pp. 674–686, doi: 10.1109/ASE63991.2025.00062.

- [5] National Institute of Standards and Technology, “Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile,” NIST AI 600-1, Gaithersburg, MD, USA, July 2024, doi: 10.6028/NIST.AI.600-1.
- [6] X. Zhang, T. Mittal, C. Bansal, R. Wang, M. Ma, Z. Ren, H. Huang, and S. Rajmohan, “FLASH: A Workflow Automation Agent for Diagnosing Recurring Incidents,” Microsoft Research, Oct. 2024. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2024/10/FLASH_Paper.pdf.
- [7] OpenAI, “Codex,” OpenAI Developers. [Online]. Available: <https://developers.openai.com/learn/codex>.