International Journal of Computing and Artificial Intelligence

E-ISSN: 2707-658X
P-ISSN: 2707-6571
www.computersciencejournals.com/ijcai
IJCAI 2025; 6(2): 129-132
Received: 07-06-2025
Accepted: 12-07-2025

V Phani Kumar
Chief Technical Officer, Terastar Networks Pvt Ltd, Hyderabad, Telangana, India

Avula Meghanatha Reddy
Managing Director, Terastar Networks Pvt Ltd, Hyderabad, Telangana, India

Corresponding Author:
V Phani Kumar
Chief Technical Officer, Terastar Networks Pvt Ltd, Hyderabad, Telangana, India

Embedding AI-driven resilience within applications: Toward native self-healing software- a review

V Phani Kumar and Avula Meghanatha Reddy

DOI: https://doi.org/10.33545/27076571.2025.v6.i2b.188

Abstract

Distributed software now operates at a complexity where infrastructure-level remediation alone does not guarantee correct system behaviour. Container orchestration can restart pods and reschedule workloads, yet applications frequently degrade due to latent logic defects, configuration regressions, dependency instability, and data drift that are invisible to coarse health probes. The dominant practice attaches observability as an external stack and relies on dashboards, playbooks, and human intervention, introducing detection latency and divorcing telemetry from the semantics embodied in code. This article advances an alternative: an AI-infused resilience model that treats observability, anomaly detection, diagnosis, and self-healing as intrinsic properties of the application layer and of the multi-application environment in which it executes. Evaluation relies on traffic replays, controlled fault injection, RCA benchmarks, and explicit cost accounting for instrumentation overhead. The overall result reframes “monitor and react” as “observe, infer, and repair,” complementing infrastructure self-healing with application-level behavioral correctness and system-level coordination. The approach is technically feasible with contemporary methods; adoption depends on disciplined instrumentation, trustworthy guardrails against false remediation, and progressive automation to build operator confidence. With advances in Operational Machine learning for AIOps, Deep learning for operational telemetry, empirical studies of distributed tracing for diagnosis in microservices with non-trivial overhead, and emergence of open datasets all have begun to standardize, enabling reproducible progress in localizing faults. Building on these foundations, the paper specifies a three-layer architecture for innate self-healing infrastructure.

Keywords: AI infused resilience, self-healing software, anomaly detection, Root cause analysis (RCA), Artificial Intelligence for IT Operations (AIOps), operational telemetry, Service level objectives (SLO’s), progressive automation

Introduction

Large-scale distributed systems incorporate hundreds of services, frequent releases, heterogeneous data stores, and third-party dependencies. Failure is expected rather than exceptional, and resilience engineering has therefore emphasized redundancy, load shedding, circuit breaking, and rapid infrastructure remediation. Container orchestration contributes automated restarts, rescheduling, and autoscaling that markedly improve availability. Yet these mechanisms address liveness rather than behavioural correctness: a restarted container can still serve wrong answers due to corrupted state, misapplied feature flags, schema drift, or downstream timeouts manifested only in particular execution paths. Treating observability as a separate platform compounds this gap. Telemetry is produced in the application but interpreted elsewhere, often by humans under time pressure. This separation introduces detection latency, loses semantic context, and makes remediation a manual, out-of-band activity that competes with release velocity and reliability goals [1]

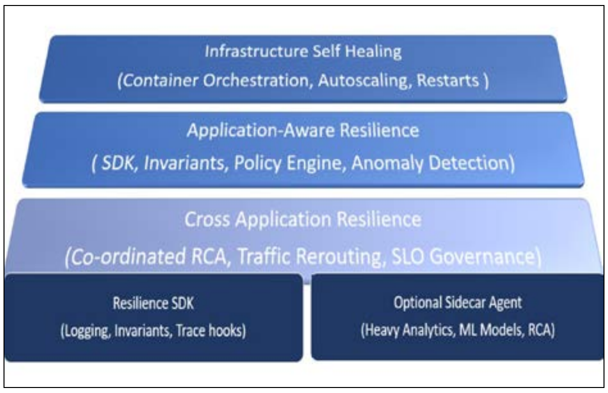
Infrastructure self-healing maintains liveness through orchestration and autoscaling. Application-aware resilience embeds structured telemetry, behavioural invariants, and machine-learning detectors that reason about control flow, output distribution and performance envelopes; these detectors drive policy-governed remediation actions such as rollback, reroute, retry and configuration repair. Cross-application resilience correlates signals across services and dependencies to contain failure impact when faults propagate beyond a single codebase. Governance is provided by service-level objectives (SLOs) and error-budget policy so that autonomous actions remain aligned with reliability targets rather than ad hoc heuristics [12]. AIOps research has advanced data-driven operations by applying machine learning to logs, metrics, and traces for failure perception, Root-cause analysis (RCA), and assisted remediation. A recent survey catalogues techniques across failure prediction and anomaly detection, multi-modal signal fusion, and automation workflows, while identifying trust, evaluation realism, and generalization as open problems [3]. In anomaly detection specifically, sequence models over logs learn normal control-flow language and flag deviations with high sensitivity, enabling detection of functional faults that static thresholds miss [4]. For performance and resource signals, advanced AI models that learn patterns across many metrics over time can spot unusual behavior more reliably than simple one-metric detectors [5]. A comprehensive review of deep learning for log analysis synthesizes these advances and highlights brittleness to log format changes, sensitivity to the preprocessing pipeline, and challenges in cross-system transfer, motivating instrumentation discipline and model maintenance as first-class concerns [6].

Surveys of RCA for microservices map methods that operate on metrics, traces, logs, and fused signals and emphasize the need for explainability and standardized evaluation to earn operator trust in semi-autonomous workflows [7]. New datasets and benchmarks address the historical deficit of common testbeds. MicroIRC shows how to pinpoint failures down to the exact service instance, with clearer evaluation methods than older system-wide approaches, while RCAEval assembles hundreds of failure cases across popular microservice benchmarks with reproducible baselines spanning metric-based, trace-based, and multi-source approaches [8, 9]. Complementary datasets such as PetShop and LEMMA-RCA extend the scope across domains and fault modes, further enabling objective comparison and ablation studies, Together, these resources make it possible to measure localization accuracy, diagnostic latency, and explanation quality under controlled faults and workload variation, an essential precondition for responsible automation [10]. Despite these advances, the dominant architecture remains layered: build the application, attach observability, and rely on external analysis and human-driven runbooks.

This article proposes AI-infused resilience as a corrective. The model embeds structured observability, machine-learning-based detection, and policy-governed remediation within applications and coordinates mitigation across applications when faults propagate. It integrates with SLOs and error-budget practice so that automated actions are gated by quantified reliability objectives rather than ad hoc rules [2]. The remainder of the paper describes an implementable architecture and evaluation method, then analyzes feasibility and risks in light of the cited literature.

Methods

The proposed architecture comprises a developer-linked Resilience SDK and an optional sidecar agent (Fig 1). The SDK provides compile-time and runtime primitives for structured logging with schema stability, metrics with explicit cardinality control, and trace hooks aligned to domain semantics so that spans represent meaningful operations rather than arbitrary call boundaries. Developers declare invariants on inputs, outputs, and state transitions; define idempotency and compensation contracts for critical sections; and identify control-flow waypoints. These declarations produce a unified event stream in which logs, metrics, and traces share correlation identifiers, enabling multi-modal feature extraction without brittle post-hoc joins. The optional agent offloads heavier analytics, such as multivariate time-series inference or graph-based RCA, and returns action proposals and explanations to the SDK.



Infrastructure Self Healing
(Container Orchestration, Autoscaling, Restarts)
Application-Aware Resilience
(SDK, Invariants, Policy Engine, Anomaly Detection)
Cross Application Resilience
(Co-ordinated RCA, Traffic Rerouting, SLO Governance)
Resilience SDK
(Logging, Invariants, Trace hooks)
Optional Sidecar Agent
(Heavy Analytics, ML Models, RCA)

Fig 1: Three-layer architecture for AI-driven self-healing application showing infrastructure, application and cross-application resilience with supporting SDK and Sidecar agent

Detection is pluggable and layered. For logs, sequence-learning models trained on normal execution learn the language of events and detect unseen or improbable transitions; DeepLog demonstrates the effectiveness of this approach for operational logs, and subsequent work generalizes the idea to richer embeddings and parsers [4]. For metrics, multivariate stochastic recurrent models such as OmniAnomaly estimate reconstruction probability and flag unlikely windows, capturing cross-signal dependence that univariate detectors miss [5]. For traces, language-model-style encoders over span sequences or graph encoders over service call graphs detect path deviations and anomalous service interactions; studies have shown that combining distributed tracing with learned representations improves sensitivity to cross-service faults compared to metrics alone [11]. The SDK exposes these detectors through stable interfaces and allows services to choose appropriate online or near-line execution plans. Online detection computes streaming features and low-latency scores in-process to reduce time-to-detect. Near-line detection in the agent uses larger windows and more computationally demanding models for deeper analysis and RCA.

Remediation is governed by a policy engine that binds conditions to actions with auditable guardrails. Policies reference SLO state and error budget so that high-disruption actions are gated by quantified reliability pressure rather than raw detector outputs [2]. A staged-autonomy model regulates trust: newly added actions begin in "observe" mode where only evidence and counterfactuals are recorded; advance to "suggest" mode where human confirmation is required; and finally graduate to "auto" for well-understood classes of faults with strong safety evidence. Actions operate at multiple scopes. Local actions include rolling back a feature flag or build, replaying a compensating transaction, draining a worker pool, invalidating a cache region, or rotating credentials. Cross-application actions operate through contracts rather than assumptions: queue requests to a degraded dependency, reduce concurrency via adaptive backpressure, or redirect traffic to a fallback implementation that satisfies essential invariants. Every action produces an explanation artifact with saliency over signals, implicated spans or metrics, detector confidence, and predicted SLO impact to support operator review and continuous improvement [7].

Closed-loop learning is built into the incident lifecycle. When a detector fires, the system records features, raw evidence, the chosen action and its scope, and outcomes such as time-to-recover, post-action error rate, and SLO minutes saved. Post-incident, this record is labeled as improved, worsened, or neutral and used to recalibrate thresholds and fine-tune models. Drift monitors watch input distributions and model residuals; when drift is detected, the system either adapts thresholds, switches to a fallback detector with known bias-variance properties, or initiates retraining using safe snapshots, consistent with observations from production drift mitigation research that not all drift is harmful and that indiscriminate retraining can degrade reliability [12, 13]. This loop reduces false positives and negatives over time and supports adaptive sampling in tracing to stay within overhead budgets documented by empirical studies [14].

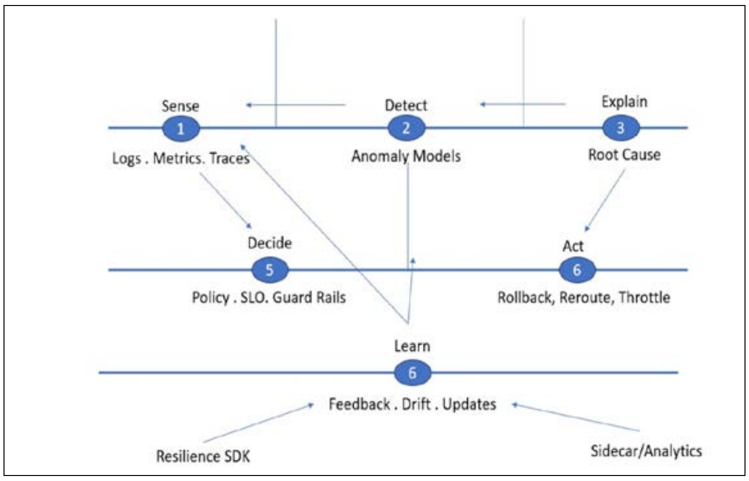


Fig 2: Flow of anomaly detection, RCA, policy-based remediation, and closed-loop learning in the proposed framework

Evaluation proceeds along four axes (Fig 2). First, detection quality is assessed via traffic replays and controlled fault injection covering latency inflation, dependency timeouts, configuration regressions, data schema drift, and logic toggles hidden behind feature flags. The principal metrics are time-to-detect and precision/recall, with ablations for individual modalities to measure contribution. Second, remediation efficacy is quantified using time-to-recover and error-budget minutes saved relative to a human-in-the-loop baseline. Third, RCA quality is evaluated on standardized datasets and benchmarks for root-cause localization, including instance-level metrics where available, using ranking metrics such as Recall@k and Mean Reciprocal Rank [8-10]. Fourth, operational overhead is measured for SDK and agent using tail-latency and CPU/memory counters, and for tracing specifically the design follows published guidance on instrumentation overhead and adaptive sampling [15]. The evaluation protocol distinguishes between local incidents resolvable within a single service and cross-application incidents requiring coordinated mitigation.

Interoperability is addressed explicitly. The SDK integrates with existing observability backends via Open Telemetry exporters and can operate in a "shim" mode that adds semantic context to existing spans and logs without replacing the underlying pipeline. Policy, detector configuration, and action catalogs are defined declaratively and stored version-controlled so that reliability decisions are reproducible and auditable. Security and safety considerations include scoping credentials for remediation actions, enforcing least privilege for cross-service operations, and requiring out-of-band human confirmation for actions that alter data at rest.

Discussion

The feasibility of embedding AI-driven resilience within applications rests on the empirical success of anomaly detection over operational telemetry and on the increasing standardization of RCA evaluation. Sequence models over logs and multivariate variational models for metrics consistently detect subtle failure modes that threshold rules overlook, provided that training data represents normal variability and that log parsing is stable [4-6]. For microservices, RCA surveys and instance-level studies demonstrate automated localization across telemetry modalities, while emerging benchmarks such as RCAEval, PetShop, and LEMMA-RCA make it possible to compare methods across datasets and domains with shared metrics and baselines [11-14]. These developments support the claim that detection and diagnosis can be made sufficiently reliable to trigger at least low-risk automated actions with bounded failure impact.

There are, however, material risks. Instrumentation has cost: Studies show that detailed tracing in microservices adds real performance costs, and full tracing is too slow for systems with strict constraints. Any practical system must therefore employ selective instrumentation and adaptive sampling, raising the bar for SDK design and configuration [14]. False positives and false negatives in detection can produce harmful or missed actions. Mitigation requires staged autonomy, SLO-aware gating, and conservative defaults that privilege containment over aggressive repair. Cold-start and drift complicate ML deployment: models trained on yesterday's "normal" can degrade as workload and software evolve, and literature on production drift mitigation recommends targeted adaptation rather than constant retraining [12, 13]. Trust and governance remain central. Surveys emphasize explainability and standardized evaluation as prerequisites for operator acceptance; explanations must cite implicated spans and metrics, quantify confidence, and connect projected action effects to SLOs rather than opaque scores [3, 7].

Positioned against external observability platforms, AI-infused resilience does not attempt to replace them. Instead, it internalizes semantics at the point of production—where invariants and intent are known—and employs the external stack for storage, visualization, collaboration, and heavy analytics. This division shortens detection and recovery paths for recurring classes of faults while preserving the established tooling and practices teams rely on. In environments with heterogeneous services and organizational boundaries, cross-application resilience must operate through explicit contracts and policy rather than implicit authority; the architecture therefore emphasizes scoped actions, minimal privileges, and auditable workflows.

The research agenda suggested by this model includes policy synthesis that learns action policies from incident history under SLO constraints; multi-modal fusion methods robust to missing telemetry and partial observability; explanations that surface minimal sufficient evidence for both detection and localization; and end-to-end benchmarks that jointly score detection, RCA, and remediation outcomes under workload variability. The availability of RCA datasets and the growing body of log and metric anomaly detection research provide a foundation for such integrated evaluation [3, 5].

Conclusion

Embedding detection, diagnosis, and repair within applications reframes resilience from an external, reactive process into a native design principle. A three-layer architecture aligns infrastructure liveness with application-level behavioral correctness and cross-application coordination. The proposed resilience SDK and sidecar agent make this approach implementable with current techniques, provided that instrumentation is disciplined, policies are SLO-aware and auditable, autonomy is staged, and evaluation is rigorous. Contemporary research and benchmarks in anomaly detection, tracing, and RCA indicate that such systems can reduce time-to-detect and time-to-recover, contain failure impact, and free engineers to focus on design and evolution rather than firefighting.

References

1. Li B, Chen T, Zhu L, Zhou A. Enjoy your observability: An industrial survey of microservice tracing and analysis. Empirical software engineering. 2022;27(1):1-11.
2. Service Level Objectives, SRE Book and Workbook resources, 2016–2020 editions.
3. Zhang L, Qian Z, He Y, Zheng Q, Ma X. A Survey of AIOps in the Era of Large Language Models. J.ACM. 2025;37(4):1-35.
4. Du M, Li F, Zheng G, Srikumar V. Deep Log: Anomaly detection and diagnosis from system logs through deep learning. Proceedings of ACM SIGSAC conference on computer and communications security. 2017; 1285-1298.
5. Su Y, Zhao Y, Niu C, Liu R, Pei W, Pei D. Omni anomaly: Robust anomaly detection for multivariate time series. Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery and data mining. 2019;2828 – 2837.
6. Landauer M, Onder S, Skopik F, Wurzenberger M. Deep learning for anomaly detection in log data: A survey. Machine Learning with Applications. 2023;12:1-19,100470.
7. Wang Tingting, Qi Guilin. A comprehensive survey on root cause analysis in (micro) services: methodologies, challenges, and trends. 2024; 10.48550/arXiv.2408.00803.
8. Zhu Y, Liu Z, Liu Q, Xie X, Zhou Y. Micro IRC: Instance-level root cause localization for microservices. Journal of systems and software. 2024;216: 112145.
9. Pham L, Zhang H, Ha H, Salim F, Zhang X. RCAEval: A benchmark for root cause analysis of microservice systems with telemetry data. ACM Proceedings, 2025; 10.48550/arXiv.2412.17015.
10. Hardt M, Orchard WR, Blobaum P, Kasiviswanathan S, Kirschbaum E. The petshop dataset finding causes of performance issues across microservices. 2023; arXiv:2311.04806v2.
11. Kohyarnejadfard I, Ghobaei-Arani M, Masdari M. Anomaly detection in microservice environments using distributed tracing data and NLP. Journal of cloud computing. 2022;11(25):1-16.
12. Mallick A, Hsieh K, Arzani B, Joshi G. Matchmaker: Data drift mitigation in machine learning for large-scale systems. Proceedings of MLSys.2022;1-25.
13. Lewis GA, Echeverría S, Pons L, Chrabaszcz J. Augur: A step toward realistic drift detection in production ml systems. ICSE: 44th International Conference on Software Engineering. 2022;37-44. 10.1145/3526073.3527590.
14. Mallick A, Hsieh K, Arzani B, Joshi G. Matchmaker: Data drift mitigation in machine learning for large-scale systems. Proceedings of MLSys. 2022;1-25.
15. Panahandeh M, Panahi S, Gandomi A. Service anomaly: Anomaly detection in microservice systems combining distributed tracing and metrics. Journal of Systems and Software. 2024;209(3):1-23.