The Eastasouth Journal of Information System and Computer Science
Vol. 2, No. 02, December 2024, pp. 275-283
ISSN: 3025-566X, DOI: 10.58812/esiscs.v2i02

Autonomous Network Troubleshooting with AIOps and Machine Learning: A Data-Driven Architecture for Correlation, Prediction, and Remediation

Mohit Bajpai
Independent researcher, Alpharetta, GA, USA

Article Info

Article history:
Received Dec, 2024
Revised Dec, 2024
Accepted Dec, 2024

Keywords:
AIOps;
Anomaly Detection;
Closed-Loop Automation;
Incident Management;
Kafka;
Kong;
Machine Learning;
MTTR;
Network Troubleshooting;
Observability;
Remedy;
Root Cause Analysis;
Telemetry

ABSTRACT

Modern networks carry business-critical traffic across data centers, cloud platforms, branch locations, APIs, telecom circuits, security gateways, and software-defined overlays. In this environment, traditional troubleshooting is no longer limited by the availability of alarms; it is limited by the volume, fragmentation, and operational interpretation of telemetry. This updated paper presents an expanded AIOps and machine learning framework for automating network troubleshooting through telemetry ingestion, data enrichment, anomaly detection, event correlation, root-cause ranking, predictive risk scoring, and governed closed-loop remediation. The paper extends the original architecture by adding data governance, model lifecycle management, explainability, human approval gates, operational KPIs, and security controls. It also introduces a telecommunications implementation scenario in which Remedy tickets, Kafka streams, Kong APIs, topology data, and AIOps model outputs are combined to reduce alert noise, accelerate diagnosis, and improve network resilience. The proposed approach is not intended to replace network engineers; rather, it converts repetitive investigation patterns into repeatable, auditable, and continuously improving operational workflows.

Corresponding Author:
Name: Mohit Bajpai
Institution: Independent researcher, Alpharetta, GA, USA
Email: bajpaimohit@gmail.com

1. INTRODUCTION

Reliable network infrastructure is now a prerequisite for digital operations. A single degradation event can affect payment systems, healthcare applications, contact centers, cloud workloads, remote workers, and customer-facing portals. The scale problem is also increasing: Cisco projected 29.3 billion networked devices and connections by 2023, including 14.7 billion machine-to-machine connections, which means that operational teams must manage a far larger and more diverse telemetry surface than in previous generations of enterprise networking [6].

At the same time, downtime has become more expensive. Uptime Institute reported in its Annual Outage Analysis 2023 that more than two-thirds of outages cost more than $100,000, and 45% of respondents placed the cost of their most recent outage between $100,000 and $1 million [5]. These figures strengthen the business case for moving from reactive troubleshooting to

predictive, automated, and evidence-driven incident management.

Artificial Intelligence for IT Operations (AIOps) provides a practical path toward this shift. Cheng et al. describe AIOps as the application of AI techniques to IT operations data, especially for incident detection, failure prediction, root-cause analysis, and automated actions [2]. In network operations, the same pattern can be applied to routing events, interface counters, firewall logs, device configuration changes, flow records, topology dependencies, service tickets, and change-management metadata.

This updated article expands the original paper by adding a stronger data model, a more complete architecture, measurable operational KPIs, security and governance controls, and a realistic telecom-oriented implementation model using Remedy, Kafka, Kong, and an AIOps-ML server cluster.

2. PROBLEM STATEMENT: WHY MANUAL TROUBLESHOOTING DOES NOT SCALE

Traditional network troubleshooting depends on engineers reviewing alarms, checking dashboards, querying logs, validating recent changes, identifying service dependencies, and manually correlating events across tools. This approach works for isolated incidents, but it becomes fragile when incidents span multiple layers such as access, aggregation, core, firewall, DNS, WAN, cloud interconnect, and application delivery.

The main operational limitations are:

a. Alert volume and duplication: one physical event can generate hundreds or thousands of downstream alerts across monitoring, ticketing, syslog, and application tools.
b. Fragmented evidence: interface counters, flow data, configuration changes, circuit inventory, topology maps, and incident history often live in different systems.
c. Slow root-cause isolation: engineers must manually decide whether symptoms are caused by congestion, routing instability, device failure, software defect, power events, or a recent configuration change.
d. Inconsistent resolution quality: two engineers may follow different investigation paths for the same event, causing variation in MTTR and escalation quality.

Limited prediction: conventional monitoring often detects threshold violations after user impact begins, rather than identifying early-warning patterns.

AIOps addresses these limitations by treating operational data as a continuously learning evidence stream. Rather than responding to each alert as an isolated message, the platform groups related signals, suppresses duplicates, identifies probable causes, ranks remediation options, and records the outcome for future learning.

3. DATA FOUNDATION FOR AIOPS-BASED NETWORK TROUBLESHOOTING

3.1 Observability and Data Ingestion

The observability layer should ingest multiple telemetry classes instead of relying on alarms alone. A mature implementation includes syslog, SNMP traps, streaming telemetry, NetFlow/IPFIX, interface counters, routing-protocol events, firewall rule-hit counts, DNS and DHCP logs, AAA logs, ticket metadata, configuration snapshots, software-version inventory, maintenance-window records, and customer/service mapping. Configuration data is especially important because many network incidents are caused by changes that are syntactically valid but operationally harmful.

Ingestion must normalize timestamps, device identifiers, interface names, severity levels, circuit IDs, customer IDs, and topology references. Without this normalization, the ML engine may treat the same device, site, or service as several unrelated entities.

Kafka can be used as a streaming backbone for high-volume telemetry, while Kong can expose governed APIs for enrichment, orchestration, and remediation calls.

3.2 Data Enrichment

Raw telemetry becomes operationally meaningful only after enrichment. For network troubleshooting, enrichment should add device role, peer group, software version, site, geography, customer/service ownership, criticality rating, topology neighbors, dependency graph position, change-window metadata, known-error references, and prior incident outcomes. This prevents the system from treating a lab access switch, a production edge firewall, and a core aggregation switch as equivalent objects simply because they produce similar counters. Table 1 organizes the key data domains that should be captured and explains how each domain supports faster troubleshooting and more accurate root-cause analysis.

Table 1. AIOps Data Domains and Troubleshooting Value

Data domain	Examples	Troubleshooting value
Telemetry	Syslog, traps, streaming metrics, interface errors, CPU, memory, flow records	Detects symptoms and early-warning deviations
Topology and inventory	CMDB, device role, circuit mapping, peer relationships, site hierarchy	Separates root events from downstream symptoms
Change data	Config diffs, maintenance windows, software upgrades, firewall policy changes	Links incidents to recent operational changes
Ticket data	Remedy incidents, categories, assignment groups, resolution notes, timestamps	Provides labels for supervised learning and MTTR analysis
Security context	AAA events, firewall denies, vulnerability records, policy violations	Identifies security-driven network degradation or isolation triggers
Business context	Service owner, customer tier, application criticality, SLA, revenue impact	Prioritizes remediation and escalation based on impact

3.3 Data Enrichment Data Quality Controls

Data quality is one of the most important success factors in AIOps. Kumar identifies data silos as a major roadblock because incomplete or disconnected datasets reduce the accuracy and actionability of AIOps outcomes [4]. In practice, the platform should enforce schema validation, duplicate removal, clock-skew correction, missing-value handling, asset identity resolution, and lineage tracking. Every model output should be traceable back to source telemetry, enrichment data, and the model version used to produce the recommendation.

4. REFERENC ARCHITECTURE

The updated architecture extends the original design by adding a formal feature store, knowledge layer, policy-controlled remediation engine, continuous learning loop, and governance controls. The goal is to produce not only an automated action, but an auditable evidence package that explains what happened, why the model ranked a cause highly, what action was taken, and whether that action improved the incident outcome.

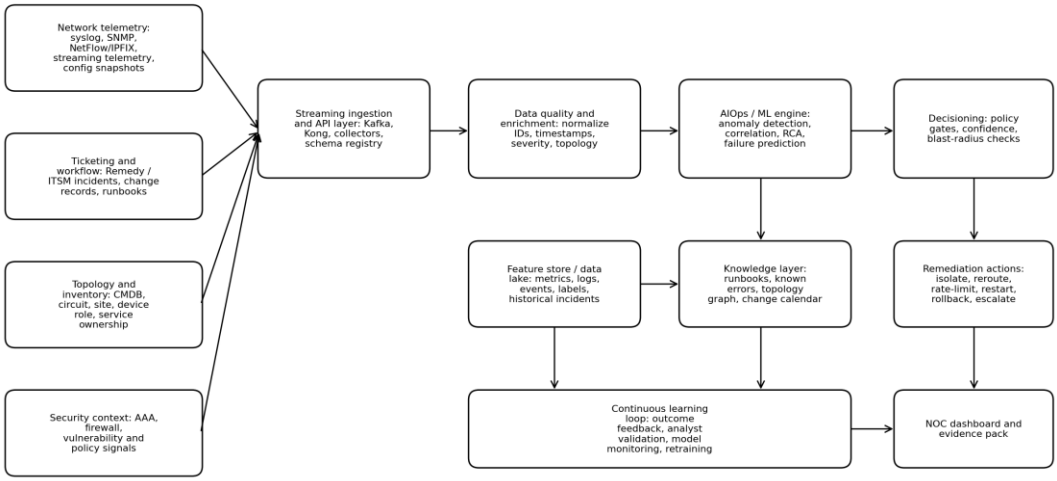

Figure 1. Updated AIOps reference architecture for autonomous network troubleshooting.

4.1 Architecture Components

Data ingestion layer: collects telemetry from network devices, collectors, ticketing systems, CMDB, topology tools, and security platforms.

Streaming and API layer: uses Kafka for event streaming and Kong for governed API access between automation components, AIOps services, and remediation tools.

Data quality and enrichment layer: normalizes identifiers, timestamps, severity, topology, and business context before model execution.

AIOps/ML engine: performs anomaly detection, correlation, root-cause analysis, failure prediction, and confidence scoring.

Feature store and data lake: stores engineered features, historical incidents, labels, and model-training datasets.

Knowledge layer: includes runbooks, known errors, topology graph, change calendar, and past resolution notes.

Decisioning and remediation engine: applies policy gates, blast-radius checks, and human approval thresholds before taking action.

Continuous learning loop: captures analyst feedback and incident outcomes to refine future model performance.

5. MACHINE LEARNING CAPABILITIES

5.1 Anomaly Detection

Anomaly detection identifies behavior that deviates from historical or peer-group baselines. In networking, this may include unusual interface discard rates, sudden latency increases, route flapping, abnormal traffic shifts, authentication failures, asymmetric routing, packet loss, tunnel fragmentation, or unusual DNS query patterns. Zhong et al. survey time-series anomaly detection methods in the AIOps domain and emphasize that real-world AIOps anomaly detection must address seasonality, noise, concept drift, multivariate dependencies, and delayed labels [3].

A practical network implementation should combine multiple detection strategies: static thresholding for well-understood limits, statistical baselining for seasonal metrics, unsupervised learning for rare patterns, supervised classification for known incident types, and topology-aware

correlation for blast-radius interpretation.

5.2 *Event Correlation and Alert Compression*

Alert compression groups related events into a smaller number of actionable incidents. For example, if a core router loses a link, the access devices, firewall sessions, voice-quality monitors, and application probes may all generate alerts. AIOps correlation should determine whether these alerts represent separate failures or symptoms of one upstream event. Techniques include temporal grouping, graph-based dependency mapping, event fingerprinting, duplicate suppression, and change-aware correlation.

5.3 *Root-Cause Ranking*

Root-cause analysis should produce a ranked list of probable causes rather than a single unexplained answer. For each probable cause, the platform should show supporting evidence such as a configuration change, topology dependency, metric deviation, related ticket history, peer-group comparison, and known-error match. This supports analyst trust and reduces the risk of blind automation.

5.4 *Predictive Risk Scoring*

Predictive modeling estimates the probability that a network component, circuit, path, or service will degrade within a defined time window. Inputs may include error-rate trends, utilization growth, device temperature, repeated flap history, software defect exposure, capacity thresholds, and historical failure sequences. The output should be a risk score linked to a recommended preventive action such as maintenance scheduling, capacity expansion, routing adjustment, or targeted observation.

5.5 *Automated and Semi-Automated Remediation*

Automated remediation should be policy-controlled. Low-risk actions may be executed automatically, such as enriching a ticket, suppressing duplicates, collecting diagnostic commands, restarting a failed monitoring collector, or rerouting traffic within approved policy. Higher-risk actions, such as firewall changes, route policy modification, failover, or configuration rollback, should require human approval unless the organization has validated the runbook through repeated safe execution. Table 2 defines the recommended remediation classes, example actions, and control levels so that automation can be introduced safely rather than applied uniformly to every incident.

Table 2. Policy-Controlled Remediation Classes for Network Automation

Remediation class	Example action	Recommended control
Informational	Attach RCA evidence and probable cause to Remedy ticket	Automatic
Diagnostic	Run show commands, collect packet-loss tests, retrieve recent config diff	Automatic with logging
Low-risk correction	Restart non-production collector or clear stale monitoring state	Automatic if confidence is high
Traffic engineering	Reroute traffic away from degraded circuit or congested path	Human approval or pre-approved policy
Security containment	Rate-limit, isolate segment, or block suspicious traffic pattern	Human approval unless active incident policy allows automation
Configuration rollback	Revert change linked to incident onset	Change-control approval and rollback validation

6. TELECOM IMPLEMENTATION SCENARIO

Consider a telecommunications provider supporting healthcare, financial, and public-sector customers. The provider experiences recurring alarms related to packet loss, voice-quality degradation, tunnel fragmentation, and intermittent circuit instability. In the manual process, the Network Operations Center reviews alarms, opens or updates Remedy tickets, checks topology, consults change records, escalates to specialized teams, and waits for additional diagnostic evidence.

In the AIOps-enabled process, Remedy tickets and network alarms are consumed by Kafka streams. Kong APIs provide controlled access to enrichment services and remediation runbooks. The AIOps-ML server cluster correlates the incoming events with historical alarms, equipment metadata, circuit inventory, maintenance windows, and topology dependencies. MongoDB or a comparable operational store retains enriched event objects, model outputs, and feedback labels. The NOC dashboard then receives a reduced incident group, a probable root cause, confidence score, recommended remediation, and supporting evidence. Table 3 compares the manual troubleshooting workflow with the AIOps-enabled workflow to show where correlation, evidence enrichment, and automation improve operational response.

Table 3. Comparison of Manual and AIOps-Enabled Troubleshooting Workflows

Manual workflow	AIOps-enabled workflow
Engineer reviews individual alarms and searches multiple tools manually.	Platform groups related alarms and creates one evidence-backed incident view.
Root cause depends heavily on engineer experience and tool familiarity.	Root-cause ranking uses topology, time correlation, change data, and historical resolutions.
Ticket notes may be inconsistent or incomplete.	The system attaches diagnostics, impacted services, probable cause, and recommendation to the ticket.
Remediation may be delayed while teams collect evidence.	Approved runbooks can collect diagnostics or execute low-risk corrective actions immediately.
Lessons learned may stay in individual knowledge.	Outcome feedback becomes training data for future correlation and prediction.

Example: A sudden increase in voice-quality complaints coincides with packet-loss alarms, tunnel fragmentation events, and a recent policy update on a WAN edge device. Instead of generating separate incidents for each symptom, the AIOps engine groups them by time, topology, and service impact. It identifies the recent configuration change as a high-likelihood cause, attaches the change ID and affected circuits to the Remedy ticket, recommends rollback or traffic reroute, and triggers a human approval workflow. After the action is completed, the platform measures whether packet loss, jitter, and related alarms return to baseline.

7. MEASUREMENT FRAMEWORK AND OPERATIONAL KPIS

An AIOps program should be evaluated through measurable outcomes rather than model accuracy alone. Accuracy is important, but operations leaders also need to know whether the platform reduces noise, accelerates triage, shortens downtime, improves escalation quality, and avoids unsafe automation. Table 4 provides the KPI framework used to evaluate whether the proposed AIOps implementation is improving incident detection, triage, resolution, and governance.

Table 4. Measurement Framework and Operational KPIs for AIOps Programs

KPI	Definition	Why it matters
Alert compression ratio	Raw alerts divided by correlated incidents	Measures noise reduction and analyst focus

KPI	Definition	Why it matters
Mean time to detect (MTTD)	Time from first abnormal signal to incident detection	Measures early-warning capability
Mean time to acknowledge (MTTA)	Time from incident creation to ownership	Measures workflow responsiveness
Mean time to resolve (MTTR)	Time from incident start to verified restoration	Measures business impact reduction
RCA precision	Percentage of top-ranked causes validated by engineers	Measures trustworthiness of recommendations
False-positive rate	Percentage of model alerts that do not represent actionable incidents	Controls alert fatigue
Automation success rate	Approved automated actions that resolve or improve the incident	Measures runbook reliability
Rollback / reversal rate	Automated actions that must be reversed	Measures automation safety
Change-related incident detection	Incidents correctly linked to recent changes	Measures change-aware troubleshooting maturity

The baseline should be calculated before automation is introduced. A strong pilot usually begins with read-only recommendations and evidence enrichment, then moves to diagnostic automation, then to limited closed-loop actions after the organization has enough validation data.

8. SECURITY, GOVERNANCE, AND EXPLAINABILITY

Network troubleshooting automation operates on sensitive data and may execute actions that affect production traffic. Therefore, the architecture must include role-based access control, API authentication, secrets management, audit logging, separation of duties, and change-management integration. NIST SP 800-53 Rev. 5 provides a useful security-control baseline for access control, audit and accountability, configuration management, incident response, and system monitoring [8].

Explainability is equally important. Engineers need to understand why the system generated a recommendation. Each recommendation should include top evidence, affected topology, impacted services, model confidence, recent changes, similar past incidents, and the proposed action. When the model is uncertain, the system should escalate rather than automate.

9. IMPLEMENTATION CHALLENGES

9.1 Data Silos and Inconsistent Identifiers

The most common implementation barrier is not the ML algorithm; it is the inability to connect the same event across monitoring, ticketing, topology, inventory, and change systems. A circuit may have ONE IDENTIFIER IN THE NOC TOOL, ANOTHER in the billing system, and another in the CMDB. Identity resolution must be treated as a core engineering capability.

9.2 Model Drift and Operational Change

Network behavior changes when new customers, routing policies, applications, cloud regions, and software versions are introduced. Models must be monitored for drift, retrained with validated labels, and tested against recent incident patterns.

9.3 Automation Risk

Closed-loop remediation can reduce MTTR, but it can also introduce risk if policy gates are weak. The safest approach is progressive autonomy: recommendation only, diagnostic automation, low-risk correction, human-approved remediation, and finally limited fully automated remediation for well-tested actions.

9.4 Human Adoption

AIOps succeeds when engineers trust the output. Trust improves when

recommendations are explainable, evidence-rich, reversible, and integrated into existing workflows instead of forcing engineers to use a separate tool.

10. BENEFITS OF THE PROPOSED APPROACH

Faster incident triage through automated evidence collection, correlation, and probable-cause ranking.

Reduced alert fatigue through duplicate suppression and topology-aware incident grouping.

Improved service reliability by detecting early-warning patterns before severe outages occur.

More consistent ticket quality through automated enrichment and standardized diagnostic attachments.

Lower operational cost by reducing repetitive manual investigation and enabling engineers to focus on complex issues.

Better governance because every automated recommendation and action is logged with evidence, confidence, and outcome feedback.

11. CONCLUSION

AIOps and machine learning can significantly improve network troubleshooting by converting fragmented telemetry into correlated, explainable, and actionable incident intelligence. The updated architecture presented in this paper combines telemetry ingestion, enrichment, anomaly detection, event correlation, root-cause ranking, predictive risk scoring, policy-based remediation, and continuous learning. This creates a practical pathway from reactive monitoring to proactive and semi-autonomous network operations.

The most important lesson is that AIOps is not a single model or dashboard. It is an operating model that depends on high-quality data, integrated workflows, governed automation, measurable KPIs, and human feedback. When implemented carefully, it can reduce alert noise, accelerate MTTR, improve service reliability, and preserve institutional knowledge across the operations lifecycle.

Future work should focus on graph-based topology reasoning, multi-modal incident analysis across metrics/logs/traces/tickets, safer closed-loop remediation, and integration of generative AI assistants that can summarize evidence and recommend runbook steps while preserving human oversight for high-risk decisions.

REFERENCES

[1] Chen, Z., Kang, Y., Li, L., Zhang, X., Zhang, H., Xu, H., Zhou, Y., Yang, L., Sun, J. C., Xu, Z., Dang, Y., Gao, F., Zhao, P., Qiao, B., Lin, Q., Zhang, D., & Lyu, M. R. (2020). Towards intelligent incident management: Why we need it and how we make it. Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. https://doi.org/10.1145/3368089.3417055

[2] Cheng, Q., Sahoo, D., Saha, A., Yang, W., Liu, C., Woo, G., Singh, M., Saverese, S., & Hoi, S. C. H. (2023). AI for IT Operations (AIOps) on Cloud Platforms: Reviews, Opportunities and Challenges. arXiv:2304.04661. https://doi.org/10.48550/arXiv.2304.04661

[3] Zhong, Z., Fan, Q., Zhang, J., Ma, M., Zhang, S., Sun, Y., Lin, Q., Zhang, Y., & Pei, D. (2023). A Survey of Time Series Anomaly Detection Methods in the AIOps Domain. arXiv:2308.00393. https://doi.org/10.48550/arXiv.2308.00393

[4] Kumar, S. (2023). Data Silos: A Roadblock for AIOps. arXiv:2312.10039. https://doi.org/10.48550/arXiv.2312.10039

[5] Uptime Institute. (2023). Annual Outage Analysis 2023. https://uptimeinstitute.com/resources/research-and-reports/annual-outage-analysis-2023

[6] Cisco. (2020). Cisco Annual Internet Report (2018-2023) White Paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[7] BMC Software. (2020). BMC Remedy ITSM Suite Overview. https://www.bmc.com/it-solutions/remedy-itsm.html

[8] National Institute of Standards and Technology. (2020). Security and Privacy Controls for Information Systems and Organizations (NIST Special Publication 800-53 Rev. 5). https://doi.org/10.6028/NIST.SP.800-53r5

[9] International Organization for Standardization. (2022). ISO/IEC 27001:2022 Information security, cybersecurity and privacy protection - Information security management systems - Requirements.
[10] Pimentel, M. A. F., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. Signal Processing, 99, 215-249. https://doi.org/10.1016/j.sigpro.2013.12.026
[11] Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. https://doi.org/10.1145/3133956.3134015
[12] Wang, S., Balarezo, J. F., Kandeepan, S., Al-Hourani, A., Gomez, K., & Rubinstein, B. (2021). Machine Learning in Network Anomaly Detection: A Survey. IEEE Access, 9, 152379-152396. https://doi.org/10.1109/ACCESS.2021.3126834