International Journal of Advanced Engineering, Management and Science (IJAEMS)
Peer-Reviewed Journal
ISSN: 2454-1311 | Vol-12, Issue-3; May-Jun, 2026
Journal Home Page: https://ijaems.com/
DOI: https://dx.doi.org/10.22161/ijaems.123.42

Strategies for Reducing Alert Fatigue and Improving Signal Quality in Enterprise Monitoring Architectures

Sushrutha Sreevathsa

Amherst, New York, USA, Service Reliability Engineer at Yahoo
Email: ssreevat@iu.edu

Received: 25 May 2026; Received in revised form: 21 Jun 2026; Accepted: 25 Jun 2026; Available online: 28 Jun 2026

Abstract *— Enterprise monitoring architectures now generate dense alert streams, yet many notifications fail to support fast incident handling. The article examines alert fatigue as an architectural and operational failure caused by redundant thresholds, fragmented telemetry, weak incident correlation, and shallow prioritization rules. The aim is to develop an analytical model for improving signal quality without using proprietary operational data. The study draws on recent academic literature covering AIOps, alert aggregation, dynamic suppression, observability, log anomaly detection, and incident management. Comparative source analysis, typologization, and conceptual synthesis distinguish rule-based, correlation-based, statistical, and machine learning-supported strategies. The results identify three intervention layers: monitor design, alert lifecycle management, and decision-oriented incident routing. The discussion develops a practical model that links suppression, aggregation, enrichment, and ticket prioritization into a staged operating sequence. The article offers guidance for enterprise teams seeking fewer alerts, stronger diagnostic value, and more reliable escalation.*

Keywords *— **Artificial Intelligence for IT Operations (AIOps), alert aggregation, alert fatigue, enterprise monitoring, signal quality***

I. INTRODUCTION

Enterprise monitoring rarely suffers from a shortage of telemetry. The harder task lies in turning telemetry into a limited set of signals that deserve operational attention. Cloud services, microservices, container platforms, distributed traces, logs, metrics, and event streams produce dense evidence about system behavior. A large share of that evidence reaches incident queues as isolated notifications. Low-value alerts then receive the same visibility as failure-bearing signals, and operational work shifts from diagnosis to sorting. The first cost appears in slower response. The second cost appears in weakened trust: engineers begin to treat the alerting layer as noise, although that layer exists to protect reliability (Yu et al., 2024; Usman et al., 2022; Voutsas et al., 2024; Tariq et al., 2025).

The aim of this article is to develop an analytical framework for reducing alert fatigue and improving signal quality in enterprise monitoring architectures. The first objective is to identify architectural and operational causes that convert monitoring coverage into alert noise. The second objective is to compare rule-based, correlation-based, statistical, and machine learning-supported strategies for filtering, suppressing, aggregating, and enriching alerts. The third objective is to formulate an implementation logic that connects monitor design, incident routing, and feedback loops into a sustainable enterprise operating model.

The novelty of the article lies in treating alert fatigue as a signal quality problem, separate from a broad observability review or a narrow discussion of human attention. The proposed framing gives priority to the path from raw events to actionable signals, then from signals to prioritized incident work. This view fits

enterprise environments where monitoring tools, ticket queues, escalation policies, service ownership, and reliability metrics mature at uneven speeds.

The hypothesis is that alert fatigue declines when monitoring architectures shift from threshold-centered notification to decision-centered signal management. In this model, an alert gains operational value only when it carries service relevance, severity, ownership, temporal relation, dependency evidence, and next-step guidance.

II. MATERIALS AND METHODS

The literature search used ACM Digital Library, IEEE Xplore, ScienceDirect, Springer-linked indexing, arXiv for fast-developing AIOps research, and publisher pages for DOI and bibliographic verification. Search strings combined "alert fatigue," "AIOps," "alert aggregation," "alert suppression," "incident management," "cloud monitoring," "observability," "root cause analysis," "log anomaly detection," and "microservices monitoring." The initial search returned 54 potentially relevant publications from 2021 to 2025. The screening removed works that focused only on cybersecurity detection without transferable alert management logic, appeared in weak publication venues, lacked technical relevance to enterprise monitoring, or discussed observability through generic tool descriptions. Ten sources remained in the final corpus. They cover intelligent alert and incident management (Yu et al., 2024), AIOps task taxonomy (Cheng et al., 2023), cloud alert fatigue mitigation (Voutsas et al., 2024), alert aggregation (Kuang et al., 2024), dynamic suppression (Bhukar et al., 2024), correlation mining for alert reduction (Yu & Zhang, 2023), monitor recommendation and monitor ontology (Srinivas et al., 2024), log anomaly detection (Landauer et al., 2023), observability in distributed edge and container-based microservices (Usman et al., 2022), and human-centered alert fatigue mitigation in security operations (Tariq et al., 2025).

Comparative analysis distinguished the operational value and limits of four strategy families: rule-based filtering, correlation and aggregation, statistical suppression, and machine learning-supported prioritization. Source analysis separated empirical findings, survey-based taxonomies, and conceptual contributions. Conceptual synthesis supported the proposed staged model of signal quality improvement. Typologization classified interventions by their place in the alert lifecycle: monitor creation, event evaluation, alert grouping, ticket prioritization, escalation, and post-incident feedback.

III. RESULTS

The first result concerns the source of alert fatigue. Recent research treats excessive alerts as a structural property of large-scale service systems, where component dependencies, repeated symptoms, and overlapping monitors create more operational messages than human teams can evaluate. A survey of intelligent alert and incident management presents alert handling as a chain that connects detection, correlation, diagnosis, assignment, and resolution, so noise created at the monitoring layer spreads into incident queues and team coordination (Yu et al., 2024). Enterprise architects need this distinction because alert fatigue begins before a ticket reaches an engineer. It emerges when the system lacks a disciplined boundary between telemetry, event, alert, and incident.

AIOps literature makes the boundary clearer. Cloud AIOps research separates incident detection, failure prediction, root cause analysis, and automated action as different problem classes, each with its own input data and evaluation criteria (Cheng et al., 2023). This taxonomy prevents a common architectural error: using one alerting method for every operational purpose. A threshold crossing, a predicted saturation pattern, a correlated alert storm, and a confirmed customer-impacting incident require different routing. Equal treatment inflates attention demand and weakens escalation discipline.

Observability research adds another layer. Distributed edge and container-based microservices require telemetry collection, storage, processing, visualization, notification, and analysis across changing service topologies (Usman et al., 2022). The monitoring architecture therefore cannot depend only on isolated metrics. Signal quality depends on whether alerts preserve service relationships, dependency paths, and runtime environment evidence. A Central Processing Unit (CPU) alert without service ownership, workload relation,

saturation history, and user-facing impact remains a weak signal. A smaller set of enriched alerts gives engineers more operational value than a large set of isolated measurements.

The second result concerns monitor design. A data-driven monitoring framework for cloud services records that teams often create monitors in an ad hoc manner based on local engineering knowledge, which produces both incomplete coverage and redundant monitors (Srinivas et al., 2024). For the first objective of this article, this point has direct relevance because alert fatigue is partly produced at monitor creation. If teams add monitors after incidents without reviewing overlap, ownership, severity thresholds, and dependency coverage, alert volume grows through organizational memory. Each incident leaves another rule, and the rule set gradually loses coherence.

This problem changes the meaning of "coverage." High monitoring coverage does not guarantee high signal quality. Coverage becomes useful when engineers connect it to service objectives, known failure modes, and actionable ownership. Enterprise teams therefore need a monitor inventory, not only a tool inventory. A monitor inventory records the condition under observation, service dependency, expected operator action, severity class, escalation target, and retirement criteria. Research on monitor ontology supports this move by showing that structured monitor attributes can be mined and used to recommend monitors based on service properties (Srinivas et al., 2024). The operational lesson is direct: teams should govern monitor creation with the same care that product teams apply to feature design.

The third result concerns alert filtering and suppression. A machine learning study on cloud monitoring introduces an alert fatigue mitigation mechanism that decides which alerts administrators should see and which alerts the system should hide, with personalization based on alert features and user experience (Voutsas et al., 2024). The contribution matters because suppression here means selective exposure, not blind deletion. Enterprise architectures need that distinction. Suppression policies should reduce low-value notification pressure while preserving auditability, delayed review, and recovery of hidden signals if later evidence changes the incident picture.

Dynamic suppression develops the same logic under changing conditions. Research on Dynamic-X-Y proposes an unsupervised policy that learns from historical alerts and events, then applies learned suppression policies at runtime to reduce false notifications in dynamic environments (Bhukar et al., 2024). Its value for the second objective lies in the problem it addresses: static suppression rules decay. Enterprise systems change through releases, scaling events, dependency shifts, and infrastructure migration. A rule that was safe during one topology period can become unsafe after service ownership or traffic shape changes. Dynamic suppression therefore needs review windows, policy drift checks, and exception handling. Without these controls, suppression can hide operational risk behind a lower alert count.

Correlation-based strategies address another failure pattern: the alert storm. Large-scale cloud systems often generate many related alerts from a smaller number of underlying causes. Knowledge-aware alert aggregation research proposes a hybrid approach that combines correlation mining with large language model reasoning over alert-related knowledge, including standard operating procedure evidence, to group alerts triggered by the same failure (Kuang et al., 2024). This contribution matters because aggregation differs from deduplication. Deduplication removes exact or near-exact repeats. Aggregation groups different symptoms that belong to the same root cause or incident episode. Enterprise monitoring gains more from aggregation when the system preserves causal hints and operational rationale.

AlertInsight reaches a similar destination through multiple correlation mining. The framework identifies correlations among event sources and aggregates correlated events into higher-level alerts, with published precision, recall, and F1 values around 0.92 to 0.93 in the evaluation (Yu & Zhang, 2023). The result supports the claim that alert reduction cannot depend only on message similarity. Two alerts can use different wording and still belong to the same incident. Two similar messages can refer to separate failures in different service domains. Multiple correlation mining helps move alert handling from text-level comparison toward episode-level interpretation.

Log anomaly detection research clarifies another layer of signal quality. A survey of deep learning methods for log anomaly detection reports that self-learning models can identify unexpected log event occurrences without manually modeled anomalous scenarios, while still facing challenges linked with preprocessing, unstable log formats, model choice, and evaluation datasets (Landauer et al., 2023). For enterprise monitoring, this creates a careful boundary. Log anomaly detection can surface signals that static thresholds miss, especially when failures appear in unstructured or semi-structured diagnostic text. It still cannot replace incident classification. A detected anomaly becomes useful only when engineers can connect it to service impact, timing, dependency, and runbook evidence.

Human-centered alert fatigue research, developed in security operations, gives a transferable warning. A review of alert fatigue mitigation in security operations organizes approaches through automation, augmentation, and human factors, which applies to enterprise monitoring teams facing high notification density and repeated triage work (Tariq et al., 2025). The operational lesson is direct. Tool improvements cannot resolve fatigue if routing logic, escalation expectations, shift design, and feedback practices remain weak. Signal quality has a human interface. Engineers judge alerts by whether the alert has helped them act in previous incidents.

The comparison across sources indicates that no single strategy family solves the problem across the full lifecycle. Rule-based filtering gives transparent control and fast implementation, but it becomes brittle in dynamic systems. Correlation-based aggregation reduces storms and supports incident grouping, but it depends on accurate temporal, topological, and semantic relationships. Statistical suppression adapts to historical patterns, but it needs safeguards against drift and hidden failure classes. Machine learning-supported approaches expand anomaly detection and prioritization, but they require governance, explainability, and evaluation against operational outcomes.

A consolidated lifecycle view follows from this comparison. The team shapes the signal before it reaches the ticket queue, enriches it before escalation, groups it before incident assignment, and reviews it after resolution. The monitoring architecture therefore needs a feedback loop from incidents back to monitors. Closed incidents reveal which alerts were useful, which arrived late, which duplicated other signals, which lacked ownership, and which should never have paged a human. This feedback loop converts incident history into alert design knowledge.

Figure 1 presents the proposed signal quality pipeline. It adapts the alert and incident lifecycle described in intelligent AIM research and aligns it with AIOps task separation and alert aggregation studies.

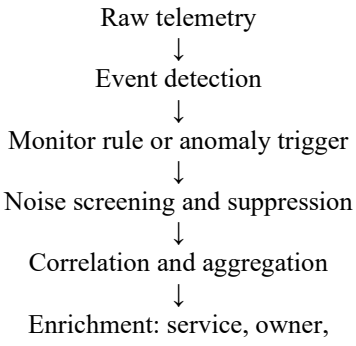

Fig.1. Signal quality pipeline for enterprise monitoring architectures (compiled by the author based on Yu et al., 2024, Cheng et al., 2023, and Kuang et al., 2024)

The figure emphasizes a lifecycle management principle: the value of an alert depends on the whole pipeline, not on the trigger alone. A technically correct trigger can still create a weak operational signal when it bypasses enrichment, grouping, and priority assignment. The same figure places ticket prioritization as a late-stage decision supported by upstream evidence. This point has direct relevance for enterprises moving from ticket-flow management to signal-driven operations. Ticket queues should receive evaluated incidents, not raw monitoring exhaust.

A comparison of three sources on alert reduction deepens this point. Voutsas et al. (2024) focus on deciding which alerts deserve administrator exposure. Bhukar et al. (2024) focus on learning suppression policies that adjust to dynamic environments. Kuang et al. (2024) focus on aggregating alert storms through correlation mining and reasoning over operational knowledge. These positions complement one another. The first reduces unnecessary presentation, the second keeps suppression adaptive, and the third consolidates

related symptoms into incident-level evidence. For the second objective of the article, filtering, suppression, and aggregation should be placed in sequence. Aggregation that starts before suppression can admit weak events into group formation. Suppression that runs without later aggregation review can hide symptoms of a wider incident. A balanced architecture makes suppression reversible and aggregation explainable.

A second comparison across Srinivas et al. (2024), Usman et al. (2022), and Landauer et al. (2023) clarifies the front end of the pipeline. Monitor recommendation research points to structured monitor design. Observability research shows the need to collect and relate metrics, logs, traces, and notification mechanisms in distributed microservices. Log anomaly detection research demonstrates the value and limits of learning from log patterns. The resulting conclusion is that signal quality begins with telemetry semantics. Enterprises cannot improve alert queues only by installing a smarter incident platform. They need better monitor metadata, dependency evidence, and log structure before downstream AIOps can operate with confidence.

The final result concerns evaluation. Alert reduction should not be measured only by fewer notifications. A system can reduce volume by suppressing relevant warnings, which improves dashboard appearance while damaging reliability. The literature points toward a broader evaluation set: precision of presented alerts, recall of failure-bearing signals, quality of grouping, timeliness of incident creation, accuracy of ownership assignment, and usefulness of the recommended action (Yu et al., 2024; Voutsas et al., 2024; Yu & Zhang, 2023). For enterprise practice, the strongest indicator is the share of alerts that lead to a meaningful operator decision. Volume matters, but decision value gives a more accurate picture of signal quality.

IV. DISCUSSION

A decision-centered model of enterprise alerting should start from a direct operating rule: no alert should enter the human response channel unless it can answer what changed, why the change affects a service, who owns the service, what confidence supports the signal, and which response path engineers should follow. Many organizations try to solve fatigue by tuning thresholds after engineers complain about noise. That intervention helps only at the surface. Stronger design starts earlier, at monitor creation, and continues through suppression, grouping, enrichment, routing, and review.

A practical implementation sequence begins with an alert inventory. Engineers examine each alert for service ownership, customer or business relevance, severity logic, dependency relation, actionability, repeat frequency, and historical incident value. Alerts with no owner or no defined response path are immature signals. They are operational reminders disguised as incident evidence. Removing them from paging channels protects the channel engineers need during real incidents.

The next step is channel separation. Informational events, anomaly candidates, warning signals, and incident-grade alerts should not share the same treatment. A warning can remain visible in dashboards or daily reviews without interrupting an on-call engineer. An anomaly candidate can enrich a correlated incident group without opening a ticket. Incident-grade alerts require immediate routing only when severity, service ownership, and response logic are clear. This separation reduces fatigue because the architecture stops treating every deviation as a demand for human action.

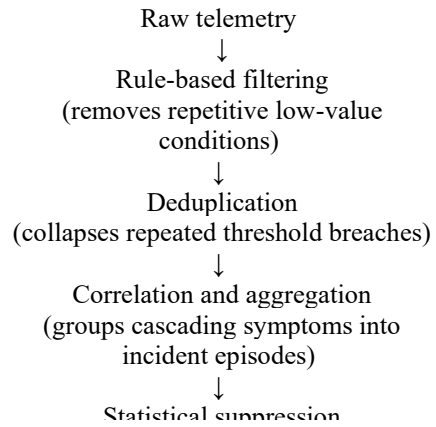

Fig.2. Layered signal reduction architecture for enterprise monitoring systems (compiled by the author based on Bhukar et al., 2024, Kuang et al., 2024, Voutsas et al., 2024, and Yu & Zhang, 2023)

Different reduction methods operate more effectively when engineers place them into a staged operational

sequence instead of applying them independently. Rule-based filtering gives teams transparent control over stable failure patterns. Statistical suppression adjusts to recurring operational behavior that static thresholds fail to capture. Correlation layers reduce alert storms by grouping related symptoms into incident-level evidence. Knowledge enrichment improves routing quality because engineers receive ownership, dependency, and runbook context before escalation begins. The sequence matters. Weak events entering aggregation layers contaminate incident grouping, while uncontrolled suppression can hide signals that later become operationally relevant.

A workable operating model contains four stages. First, monitor hygiene removes obsolete monitors, duplicate rules, orphaned alerts, and signals without defined actions. Second, signal shaping applies filtering, suppression, and severity normalization before alerts enter incident queues. Third, incident formation groups related symptoms, enriches them with ownership and dependency data, and assigns priority. Fourth, learning closes the loop by feeding incident outcomes back into monitor design.

This model needs measurement. Alert fatigue cannot be managed through raw alert counts alone. A lower count can conceal missing coverage. A higher count can be acceptable during a major release window if alerts are grouped and routed correctly. Better metrics focus on signal quality: percentage of alerts that result in useful operator action, percentage of incident tickets created from correlated groups, false-positive rate by service, duplicate-alert ratio, alert-to-incident conversion rate, time from first signal to correct owner, escalation reversal rate, and post-incident monitor changes completed within a defined review period. Table 1 presents a compact measurement framework. It connects architectural interventions with observable operational indicators, without relying on confidential dashboards or employer-specific metrics.

Table 1. Signal quality measurement framework for alert fatigue reduction (compiled by the author based on Cheng et al., 2023, Landauer et al., 2023, Srinivas et al., 2024, Tariq et al., 2025, Usman et al., 2022, and Yu et al., 2024)

Measurement area	Indicator	What it reveals	Architectural response
Actionability	Share of alerts leading to a documented operator decision	Whether alerts support work or merely occupy attention	Redesign alert conditions and response text
Noise concentration	Duplicate-alert ratio by service and monitor type	Whether repeated symptoms dominate the queue	Apply deduplication and aggregation
Incident formation	Alert-to-incident conversion rate	Whether monitoring produces incident-grade evidence	Separate events, warnings, and incident alerts
Ownership accuracy	Time from first alert to correct service owner	Whether routing reflects service topology	Strengthen service catalog and ownership metadata
Suppression safety	Reopened or later-relevant suppressed alerts	Whether suppression hides useful signals	Add reversible suppression and audit review
Diagnostic depth	Alerts enriched with dependency, severity, and runbook evidence	Whether engineers receive enough material for action	Attach topology, runbook, and incident history
Learning loop	Post-incident monitor changes completed after review	Whether incidents improve future signal quality	Formalize monitor retrospectives
Human load	Pages per on-call period by severity class	Whether paging remains cognitively sustainable	Reserve paging for incident-grade signals

The measurement framework shifts attention from alert volume to alert yield. Yield means the proportion of signals that help engineers make the next operational decision. This distinction prevents a common failure mode: presenting alert reduction as success while the team loses early warnings. The proposed metrics treat suppression as safe only when suppressed signals remain reviewable and when later incidents are checked against suppression decisions.

Implementation should proceed in controlled stages. A pilot can begin with one service family where ownership is clear and incident history is sufficient. The team maps monitors, classifies alerts by actionability, identifies duplicates, and separates dashboard-only warnings from paging alerts. Next, engineers introduce correlation rules for repeated incident patterns. Statistical suppression follows after baseline review confirms which deviations are harmless. Machine learning-supported ranking can then be tested on historical queues before it affects live escalation.

Human accountability remains central. Automated suppression, clustering, and priority scoring should change what reaches engineers, but they should not erase responsibility for alert policy. Each service needs an owner for monitor definitions, severity classes, and runbook evidence. Platform teams can own common tooling, but service teams should own the meaning of their alerts. Without that division, the organization creates a centralized alerting layer that lacks domain knowledge.

The model has limits. It does not propose full automation of incident response. It does not claim that machine learning can resolve ambiguous operational judgment. It assumes that enterprise teams maintain service catalogs, dependency maps, and post-incident review routines. If these foundations are absent, alert fatigue reduction should start with ownership and metadata before advanced AIOps. A poorly described service graph weakens aggregation. An outdated runbook weakens knowledge-enriched reasoning. A ticket queue with vague resolution notes weakens feedback into monitor design.

For engineering teams with site reliability and network operations backgrounds, the applied value lies in the transition from ticket-flow management to signal management. Ticket-flow management counts items, routes queues, and accelerates closure. Signal management asks whether the item deserved attention before it became work. The difference affects architecture. A ticket platform can organize work after noise enters the system. A signal-quality architecture reduces noise before engineers must spend time on it.

V. CONCLUSION

Enterprise alert fatigue originates in the gap between telemetry growth and decision-grade signal formation. Monitoring coverage becomes harmful when raw events, threshold deviations, anomaly candidates, and incident-grade alerts pass through the same notification path. The review confirms that signal quality depends on monitor design, alert lifecycle controls, correlation logic, ownership metadata, and post-incident feedback.

The comparison of reduction strategies shows that rule-based filtering, deduplication, dynamic suppression, correlation-based aggregation, log anomaly detection, and machine learning-supported prioritization solve different parts of the problem. Their value increases when teams order them as a pipeline: monitor hygiene first, suppression and filtering second, aggregation and enrichment third, ticket prioritization fourth, learning feedback last. The hypothesis is supported: alert fatigue declines when monitoring architectures shift from threshold-centered notification to decision-centered signal management. This conclusion rests on alignment between published AIOps taxonomies, alert aggregation studies, cloud alert fatigue mitigation research, monitor recommendation work, and observability literature.

The proposed implementation model gives enterprise teams a practical route for reducing noise without weakening reliability: classify alerts by actionability, protect paging channels, enrich signals with ownership and dependency evidence, evaluate suppression safety, and measure alert yield through operator decisions instead of raw volume alone.

REFERENCES

[1] Bhukar, K., Kumar, H., Mahindru, R., Arora, R., Nagar, S., Aggarwal, P., & Paradkar, A. (2024). Dynamic alert suppression policy for noise reduction in AIOps. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice* (pp. 178–188). https://doi.org/10.1145/3639477.3639752

[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. https://doi.org/10.48550/arXiv.2304.04661

[3] Kuang, J., Liu, J., Huang, J., Zhong, R., Gu, J., Yu, L., Tan, R., Yang, Z., & Lyu, M. R. (2024). Knowledge-aware alert aggregation in large-scale cloud systems: A hybrid approach. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice* (pp. 369–380). https://doi.org/10.1145/3639477.3639745

[4] Landauer, M., Onder, S., Skopik, F., & Wurzenberger, M. (2023). Deep learning for anomaly detection in log data: A survey. *Machine Learning with Applications, 12*, 100470. https://doi.org/10.1016/j.mlwa.2023.100470

[5] Srinivas, P., Husain, F., Parayil, A., Choure, A., Bansal, C., & Rajmohan, S. (2024). Intelligent monitoring framework for cloud services: A data-driven approach. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice* (pp. 381–391). https://doi.org/10.1145/3639477.3639753

[6] Tariq, S., Baruwal Chhetri, M., Nepal, S., & Paris, C. (2025). Alert fatigue in security operations centres: Research challenges and opportunities. *ACM Computing Surveys, 57*(9), 1–38. https://doi.org/10.1145/3723158

[7] Usman, M., Ferlin, S., Brunstrom, A., & Taheri, J. (2022). A survey on observability of distributed edge & container-based microservices. *IEEE Access, 10*, 86904–86919. https://doi.org/10.1109/ACCESS.2022.3193102

[8] Voutsas, F., Violos, J., & Leivadeas, A. (2024). Mitigating alert fatigue in cloud monitoring systems: A machine learning perspective. *Computer Networks, 250*, 110543. https://doi.org/10.1016/j.comnet.2024.110543

[9] Yu, M., & Zhang, X. (2023). AlertInsight: Mining multiple correlation for alert reduction. *Computer Systems Science and Engineering, 46*(2), 2447–2469. https://doi.org/10.32604/csse.2023.037506

[10] Yu, Q., Zhao, N., Li, M., Li, Z., Wang, H., Zhang, W., Sui, K., & Pei, D. (2024). A survey on intelligent management of alerts and incidents in IT services. *Journal of Network and Computer Applications, 224*, 103842. https://doi.org/10.1016/j.jnca.2024.103842