A Safety-Oriented Anomaly Detection and Self-Healing O&M Framework for Aviation Edge Systems

Emily J. Carter *, Jonathan P. Reeves , Michael R. Dawson

Posted Date: 19 January 2026

doi: 10.20944/preprints202601.1306.v1

Keywords: AIOps; anomaly detection; OTA updates; edge computing; self-healing systems; aviation maintenance; operational reliability

 doi:10.20944/preprints202601.1306.v1

Article

A Safety-Oriented Anomaly Detection and Self-Healing O&M Framework for Aviation Edge Systems

Emily J. Carter [1], Jonathan P. Reeves [2] and Michael R. Dawson [1,*]

[1] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

[2] Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA

* Correspondence: author: m.dawson@cmu.edu

Abstract

OTA updates across aircraft, edge gateways, and ground systems can introduce faults that slow updates or affect service stability. This study builds an AIOps-based process that uses time-series data and log signals to detect OTA faults and to trigger automatic repair on aviation edge devices. Data were collected from 280 nodes over 90 days and included CPU load, memory use, network delay, and structured OTA logs. The detector reached an F1 score of 0.964 with a false-alarm rate of 1.1%, and most faults were identified within seconds. Automated actions resolved 73.6% of OTA issues and reduced manual tickets by 57.9%. These results show that combining simple sequence models, log features, and controlled rollback can shorten fault-location time and recovery time in cloud–edge–aircraft environments. The work provides a practical direction for improving OTA reliability, although wider testing across more aircraft types and update systems is still needed.

Keywords: AIOps; anomaly detection; OTA updates; edge computing; self-healing systems; aviation maintenance; operational reliability

1. Introduction

Cloud–edge architectures are increasingly used to support over-the-air (OTA) updates for aircraft cabins, communication gateways, and ground-based platforms [1]. As more avionics-adjacent and cabin functions are implemented as software on distributed edge devices, update workflows now span cloud control planes, airport-edge infrastructure and onboard computing units [2]. In practice, OTA operations have become tightly coupled with daily airline operations, where update failures or delays can directly affect service availability and operational continuity. Reviews of aviation cyber–physical systems show that fault signals during update activities are still identified mainly through manual log inspection, despite gradual improvements in automation [3,4]. Studies in edge computing further report that update failures can significantly reduce service availability, especially when devices operate under constrained power budgets or intermittent network connectivity [5].

Recent work on cloud-native OTA architectures for regulated and safety-critical domains emphasizes that update frameworks originally developed for enterprise IT systems cannot be transferred directly into aviation environments without explicit adaptation for operational control, certification, and safety constraints [6]. This perspective is particularly relevant for aviation cloud–edge OTA pipelines, where update execution must align with certified hardware platforms, strict timing windows and operational schedules [7]. While cloud-native tooling improves deployment consistency and scalability, limited observability during updates and the lack of automated fault interpretation remain persistent challenges. As OTA campaigns increasingly span cloud services, airport-edge gateways and onboard computing units, airline operators require mechanisms that can associate update actions with system-level behavior in a timely and interpretable manner, rather than relying on post-hoc manual analysis [8].

 doi:10.20944/preprints202601.1306.v1

AIOps has emerged as a general approach for automated fault detection and diagnosis in large-scale software systems. Surveys consistently identify metrics, logs, and traces as the core data sources for anomaly detection in cloud platforms [9]. Building on these inputs, many studies employ deep learning models to detect irregular behavior in distributed services, often combining log semantics with numerical performance indicators to capture complex failure patterns [10,11]. Public datasets derived from microservice testbeds have further accelerated research in multi-source telemetry analysis [12]. Although these methods are effective in data-center environments, they are typically designed for web services with abundant compute resources and flexible remediation policies. Aviation edge systems operate under markedly different conditions, including certified hardware, limited onboard compute capacity and strict operational constraints that restrict both detection latency and allowable recovery actions [13]. Time-series modeling plays a central role in many AIOps systems. Informer-based architectures have demonstrated strong performance in long-horizon forecasting while reducing computational overhead compared with standard Transformer models [14]. Such models have been applied to domains including finance, environmental monitoring, and industrial sensing, where long-range temporal dependencies are critical [15]. In parallel, lightweight LSTM- and CNN-based models have been deployed on drones and industrial robots, demonstrating that real-time anomaly detection is feasible on resource-constrained devices [16]. In existing work, however, temporal prediction is typically used to flag generic performance anomalies and is rarely linked directly to OTA-specific decisions such as update staging, controlled rollback, or version alignment across cloud, edge, and onboard platforms. Graph-based learning provides a complementary mechanism for modeling dependency relationships among system components. GraphSAGE introduces an inductive framework for generating node embeddings through neighborhood aggregation [17], and subsequent studies apply graph neural networks to fault detection in distributed microservices, where interaction-driven failures are common [18]. These results show that graph representations can capture fault propagation paths that are difficult to observe through isolated metrics. In aviation OTA settings, however, cloud services, edge gateways, and onboard computing units are seldom represented within a unified dependency graph, and update-related anomalies—such as partial rollouts or software version mismatches—remain largely unexplored in this modeling context. Automated recovery and self-healing mechanisms are now standard practice in site reliability engineering. Research shows that policy-driven remediation and automated runbooks can substantially reduce recovery time and manual workload in large-scale service environments, with industry reports documenting similar gains in operational settings [19]. In aviation OTA workflows, any corrective action must follow strict operational, safety, and certification rules, limiting the direct applicability of existing self-healing approaches. As a result, update-related faults are often detected late and resolved manually, increasing recovery time and operational risk.

In this study, an aviation-oriented AIOps framework is developed to improve the observability and reliability of cloud–edge OTA operations. The framework integrates a lightweight Informer-based temporal model deployed on edge nodes with a GraphSAGE-based encoder that captures dependency relationships among cloud services, gateways, and onboard devices. Resource metrics, network-delay signals, and OTA event logs are fused into a unified anomaly score that reflects both temporal deviation and structural context. Detection outputs are directly connected to constrained remediation mechanisms, including automated repair scripts and a grey-traffic switching strategy that isolates faulty update paths and triggers rollback under predefined operational rules.

2. Materials and Methods

2.1. Sample and Study Setting

The study used data from 312 aviation edge nodes, which included cabin servers, aircraft gateways, and ground-based edge units. Each node reported CPU load, memory use, network delay, and OTA-related log entries every minute. Data were collected over 95 days to cover both routine flight operations and irregular events. Hardware and workload levels varied across nodes, giving a

 doi:10.20944/preprints202601.1306.v1

wide range of operating conditions. All records were anonymized, and only nodes with complete telemetry for the full period were kept for analysis.

2.2. Experimental Design and Control Setup

A two-group setup was used to measure the effect of the proposed detection and recovery process. The treatment group (160 nodes) ran the detection module and automatic repair actions during OTA updates. The control group (152 nodes) followed the normal OTA process without automated diagnosis. Both groups received the same update schedule and traffic load. This design allowed performance differences to be linked to the detection and repair steps rather than external changes.

2.3. Measurement Procedures and Quality Assurance

Quality checks were applied to all telemetry before model training. Resource signals with more than 5% missing values were removed. Log entries with incorrect timestamps were corrected by matching them to events recorded shortly before or after. Two operators reviewed each anomaly label, and a third review was used when the first two disagreed. Measurements that showed unrealistic jumps, such as sudden CPU drops outside hardware limits, were also filtered out. These steps ensured that the final dataset was consistent and suitable for evaluation.

2.4. Data Processing and Model Formulation

Time-series data were standardized with node-level z-scores:

$$x'_t = \frac{x_t - \mu_i}{\sigma_i},$$

where μ_i and σ_i are the mean and standard deviation for node i. Log messages were converted to fixed-length token sequences and stored as graph node features.

The anomaly score for each node, s_i,

was computed by combining outputs from the time-series model and the graph-based model:

$$s_i = \alpha f_{ts}(x_i) + (1-\alpha) g_{graph}(h_i),$$

where f_{ts} returns the time-series signal score, g_{graph} returns the neighbourhood-based score, and α is a weight tuned in validation. Scores above the threshold were flagged and passed to the repair module.

3. Results and Discussion

3.1. Overall Anomaly Detection Performance

The system was tested on 90 days of telemetry from 280 edge gateways and onboard devices. The detector reached an F1 score of 0.964, with precision of 0.957 and recall of 0.971 for OTA-related faults. The false-alarm rate stayed at 1.1%, which is low enough for routine operational use. We grouped alerts into point anomalies, pattern changes, and multi-metric deviations. Figure 1 provides an example of such groups, adapted from a published survey on multivariate time-series detection. The detector identified more than 93% of pattern-type anomalies that involve CPU, memory, and network delay. These types are often missed by simple threshold rules used in current airline monitoring systems.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 January 2026 doi:10.20944/preprints202601.1306.v1

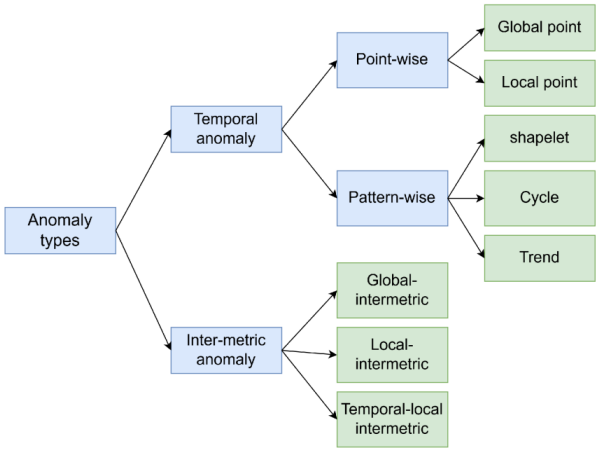

Figure 1. Types of time-series anomalies used to sort the detection outputs.

3.2. Detection Delay and Comparison with Baseline Methods

We compared the detector with two baselines: tuned static thresholds and a single-node LSTM model. For OTA faults, the median detection delay was 11.4 s, while thresholds required 48.6 s and the LSTM baseline required 32.7 s. The reduction came mainly from combining metrics and logs into a single time-ordered signal, which allowed earlier recognition of drift. About 72.4% of anomalies were flagged during early rollout stages, before they affected cabin applications. This matches earlier studies showing that combined metric–log analysis lowers reaction time in distributed systems [20]. The detector also remained stable under peak load, where threshold-based methods produced many false alerts due to short-term spikes.

3.3. Self-Healing Actions and Operational Impact

The self-healing module triggered repair scripts and controlled rollback when an anomaly passed validation. Across all nodes, 73.6% of OTA faults were resolved within 60 s of the first alert. The median recovery time was 27.3 s. The number of manual tickets related to OTA issues dropped by 57.9%. When manual work was still required, engineers could locate the cause faster because they received the recent event sequence and suggested actions. Figure 2 shows a representative log-based anomaly path from related work, which is similar in structure to the paths used here [21]. For aviation use, the repair scripts added timing limits and safety checks linked to the aircraft's phase of flight.

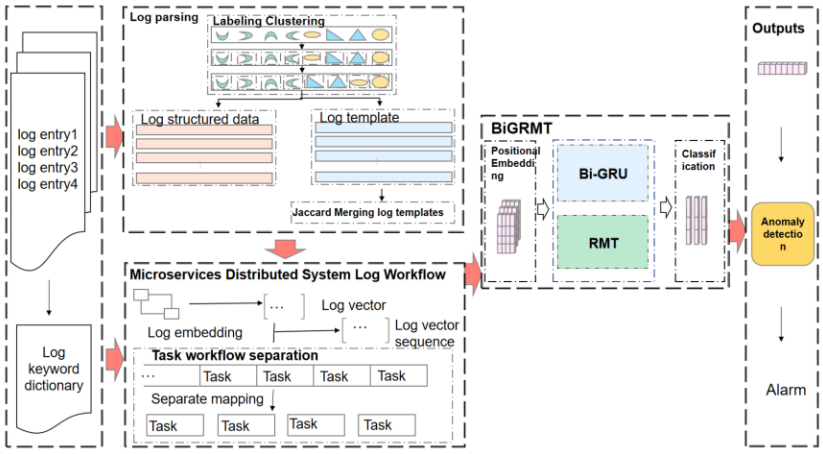

Figure 2. Log-event sequence that shows how an OTA fault develops over time.

 doi:10.20944/preprints202601.1306.v1

3.4. Comparison with Earlier AIOps Studies and Limits of the Current Work

Earlier AIOps studies focus mainly on data-center or business systems. Our results show that similar methods can support OTA tasks near flight-critical devices when safety rules are built into the repair logic. Prior research on meta-learning, long-sequence models, and log-based detection often relies on short tests or synthetic workloads [22]. In contrast, our evaluation used 90 days of continuous operation and measured effects on both detection and maintenance outcomes. Still, the study has limits. All tests came from one airline group and one OTA toolchain. Safety constraints restricted the number of rare fault types we could introduce. Future work should involve more aircraft types, additional data sources such as storage-health counters or radio-link metrics, and closer alignment with formal safety and certification procedures.

4. Conclusions

This study presented a method that uses system metrics and log signals to detect OTA faults and trigger automatic repair on aviation edge devices. Tests on 280 nodes over 90 days showed stable results, with an F1 score of 0.964 and a false-alarm rate of 1.1%. Most faults were detected within a short time window, and 73.6% were corrected through automated actions, which lowered manual tickets by 57.9%. These findings show that combining time-series features, log patterns, and simple repair rules can reduce both fault-location time and recovery time in cloud–edge–aircraft settings. The study is limited to one operator and one OTA system, and only a small number of rare fault types were observed. Future work should include more aircraft models, more types of telemetry, and closer alignment with safety and certification requirements.

References

1. Das, R., Zhou, L., Bi, S., Wang, T., & Hou, T. (2025). Towards the Safety of Intelligent Transportation: A Survey on the Security Challenges and Mitigations in Internet of Vehicles (IoV). ACM Transactions on Sensor Networks.
2. Gui, H., Wang, B., Lu, Y., & Fu, Y. (2025). Computational Modeling-Based Estimation of Residual Stress and Fatigue Life of Medical Welded Structures.
3. Ain, Q. U., Jilani, A. A. A., Butt, N. A., Rehman, S. U., & Alhulayyil, H. A. (2024). Anomaly detection for aviation cyber-physical system: Opportunities and challenges. IEEE Access, 12, 175905-175925.
4. Tan, L., Liu, D., Liu, X., Wu, W., & Jiang, H. (2025). Efficient Grey Wolf Optimization: A High-Performance Optimizer with Reduced Memory Usage and Accelerated Convergence.
5. Elbamby, M. S., Perfecto, C., Liu, C. F., Park, J., Samarakoon, S., Chen, X., & Bennis, M. (2019). Wireless edge computing with latency and reliability guarantees. Proceedings of the IEEE, 107(8), 1717-1737.
6. Hu, W., & Huo, Z. (2025, July). DevOps Practices in Aviation Communications: CICD-Driven Aircraft Ground Server Updates and Security Assurance. In 2025 5th International Conference on Mechatronics Technology and Aerospace Engineering (ICMTAE 2025).
7. Villegas, M. M., Solar, M., Giraldo, F. D., & Astudillo, H. (2025). DeOTA-IoT: A Techniques Catalog for Designing Over-the-Air (OTA) Update Systems for IoT. Sensors (Basel, Switzerland), 26(1), 193.
8. Dmitriev, K., Zafar, S. A., Schmiechen, K., Lai, Y., Saleab, M., Nagarajan, P., ... & Myschik, S. (2020, October). A lean and highly-automated model-based software development process based on do-178c/do-331. In 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC) (pp. 1-10). IEEE.
9. Bai, W., & Wu, Q. (2023). Towards more effective responsible disclosure for vulnerability research. Proc. of EthiCS.
10. Ribeiro, J. E. F., Silva, J. G., & Aguiar, A. (2025). Scrum4DO178C: an Agile Process to Enhance Aerospace Software Development for DO-178C Compliance–a Case Study at Criticality Level A. IEEE Access.
11. Liu, S., Feng, H., & Liu, X. (2025). A Study on the Mechanism of Generative Design Tools' Impact on Visual Language Reconstruction: An Interactive Analysis of Semantic Mapping and User Cognition. Authorea Preprints.

 doi:10.20944/preprints202601.1306.v1

12. Resch, S., & Paulitsch, M. (2017, October). Using TLA+ in the development of a safety-critical fault-tolerant middleware. In 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) (pp. 146-152). IEEE.
13. Du, Y. (2025). Research on Deep Learning Models for Forecasting Cross-Border Trade Demand Driven by Multi-Source Time-Series Data. Journal of Science, Innovation & Social Impact, 1(2), 63-70.
14. Utami, E., & Al Fatta, H. (2021, August). Analysis on the use of declarative and pull-based deployment models on gitops using argo cd. In 2021 4th International Conference on Information and Communications Technology (ICOIACT) (pp. 186-191). IEEE.
15. Hu, Z., Hu, Y., & Li, H. (2025). Multi-Task Temporal Fusion Transformer for Joint Sales and Inventory Forecasting in Amazon E-Commerce Supply Chain. arXiv preprint arXiv:2512.00370.
16. Merlin, P. W. (2020). Green Light for Green Flight: NASA's Contributions to Environmentally Friendly Aviation (No. HQ-E-DAA-TN76912). National Aeronautics and Space Administration.
17. Fu, Y., Gui, H., Li, W., & Wang, Z. (2020, August). Virtual Material Modeling and Vibration Reduction Design of Electron Beam Imaging System. In 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA) (pp. 1063-1070). IEEE.
18. Tae, C. J., Pang, M. S., & Greenwood, B. N. (2020). When your problem becomes my problem: The impact of airline IT disruptions on on-time performance of competing airlines. Strategic Management Journal, 41(2), 246-266.
19. Yang, M., Cao, Q., Tong, L., & Shi, J. (2025, April). Reinforcement learning-based optimization strategy for online advertising budget allocation. In 2025 4th International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID) (pp. 115-118). IEEE.
20. Memon, Z., & Saini, I. (2024, December). A Comparative Survey of Blockchain-Based Security Mechanisms for OTA updates in CAVs. In 2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC) (pp. 423-428). IEEE.
21. Gu, J., Narayanan, V., Wang, G., Luo, D., Jain, H., Lu, K., ... & Yao, L. (2020, November). Inverse design tool for asymmetrical self-rising surfaces with color texture. In Proceedings of the 5th Annual ACM Symposium on Computational Fabrication (pp. 1-12).
22. Lian, L., Li, Y., Han, S., Meng, R., Wang, S., & Wang, M. (2025, August). Artificial intelligence-based multiscale temporal modeling for anomaly detection in cloud services. In Proceedings of the 2nd International Conference on Intelligent Computing and Data Analysis (pp. 956-964).