INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY EVOLUTIONARY RESEARCH

An Automated AI-Driven Monitoring and Observability Framework for Cloud-Based Data Pipelines by Software Defect Prediction Research

Vineeth Kumar Reddy Mittamidi
Data Engineer, MetLife, Cary, NC, USA

* Corresponding Author: **Vineeth Kumar Reddy Mittamidi**

Article Info

P-ISSN: 3051-3502
E-ISSN: 3051-3510
Volume: 05
Issue: 01
January - June 2024
Received: 11-12-2023
Accepted: 13-01-2024
Published: 15-02-2024
Page No: 109-112

Abstract
Cloud-based data pipelines increasingly power machine learning, analytics, and decision automation, yet their reliability is constrained by two coupled failure modes: (i) software defects in orchestration and processing code, and (ii) silent data quality degradation that propagates to downstream features and predictions. This paper proposes an automated, AI-driven monitoring and observability framework that unifies telemetry (logs, metrics, traces), pipeline metadata, and defect prediction into a closed-loop operations capability. The framework integrates standardized instrumentation and collection, an analytics layer for anomaly detection and drift monitoring, a defect prediction subsystem that learns fault-proneness signals from historical builds and runtime indicators, and a root-cause analysis (RCA) component that leverages dependency graphs derived from traces and lineage. A governance layer encodes service-level objectives (SLOs), data quality expectations, and remediation policies to reduce mean time to detect (MTTD) and mean time to recover (MTTR) while improving trust in pipeline outputs. We present a reference architecture, feature schema, and evaluation protocol, and we demonstrate how the approach supports actionable observability across batch and streaming workloads, enabling proactive interventions before defects and data issues impact consumers.

DOI: https://doi.org/10.54660/IJMER.2024.5.1.109-112

Keywords: AIOps, observability, data pipelines, defect prediction, data quality, telemetry, drift detection, root cause analysis

1. Introduction

Modern enterprises depend on cloud-based batch and streaming pipelines to deliver fresh, consistent, and complete data to analytics and ML workloads. As these pipelines grow in scale and heterogeneity, operational signals become fragmented across orchestration layers, compute engines, storage services, and downstream consumers, complicating incident response and reliability engineering. [24]

Traditional monitoring focuses on predefined metrics and threshold alarms. Observability expands this by enabling operators to explain novel failure modes through rich telemetry and context, including distributed traces, logs, and structured events. For data pipelines, observability must extend beyond infrastructure health to include data quality, schema evolution, lineage, and downstream impact. [3]

In practice, pipeline incidents frequently originate from latent software defects (e.g., retry storms, incorrect join logic, concurrency bugs) or from data issues (e.g., null explosions, skewed distributions, source outages) that escape unit testing and surface only after deployment. Treating observability and software quality as separate concerns leaves blind spots that delay detection and increase remediation cost. [21]

This paper contributes an integrated framework that (a) standardizes telemetry collection, (b) learns predictive signals for defects and pipeline failures, (c) performs drift-aware data quality monitoring, (d) conducts RCA using dependency and lineage graphs, and (e) automates policy-governed remediation. The design aligns with practical operational constraints such as noisy alerts, incomplete labels, and heterogeneous tooling. [25]

2. Background and Problem Statement

Cloud data pipelines typically include ingestion (APIs, CDC, file drops), processing (ETL/ELT, streaming operators), storage (object stores, lakehouses, warehouses), and serving layers. Failures can be transient (network glitches) or systematic (logic defects), and may manifest as missed schedules, late data, corrupted outputs, or degraded model performance. [12]

Defect prediction uses historical software artifacts and metrics to estimate fault-proneness before release. In pipeline systems, defect prediction is especially valuable because deployment cadence is high and rollback may be costly when stateful jobs and backfills are involved. [6]

Data quality failures are frequently silent: pipelines may complete successfully while producing semantically incorrect or statistically shifted data. Drift detection and automated validation aim to detect these issues without relying on handcrafted rules for every dataset and partition. [23]

3. Related Work

AIOps research studies how machine learning can reduce alert noise, detect anomalies, and assist troubleshooting by mining operational data at scale. Surveys highlight the need for robust data integration and human-centered interpretability. [1]

Observability for ML pipelines emphasizes end-to-end visibility across ETL, feature generation, training, and serving. Database and systems communities propose metadata-centric approaches to connect data quality to downstream model behavior. [11]

Recent microservice studies combine multi-source telemetry to unify anomaly detection and RCA, and demonstrate that dependency modeling can improve localization accuracy compared with trace-only methods. [17]

Interpretable RCA methods seek to align causal inference with operator mental models, enabling faster diagnosis and higher trust in automated recommendations. [14]

Data pipeline quality research catalogs factors that influence reliability and identifies root causes such as schema changes, upstream data errors, and orchestration misconfigurations, motivating systematic monitoring of both process and data. [15]

4. Proposed Framework

Reference architecture: (i) Instrumentation Layer, (ii) Telemetry Collection and Normalization, (iii) Observability Storage, (iv) Analytics and Decision Engine, and (v) Action and Governance Layer. The framework supports both batch (scheduled DAGs) and streaming (continuous) jobs through unified event schemas and adapters. [13]

Instrumentation adopts vendor-neutral telemetry signals (traces, metrics, logs) and enriches them with pipeline context (job identifiers, dataset versions, partitions, SLA windows). This enables correlation between code-level errors and data-level symptoms. [5]

Lineage and dependency context is captured through run, job, and dataset events. This improves impact analysis by linking failures to affected downstream datasets and consumers, which is essential for prioritization and rollback decisions. [10]

4.1. Framework Modules

Table 1:

Module	Inputs	Outputs / Actions
Telemetry Normalizer	Logs, metrics, traces; job metadata	Unified events; correlation IDs
Data Quality Monitor	Schema stats; distribution summaries; drift features	Alerts; quality scores; quarantine triggers
Defect Predictor	Code metrics; change history; runtime failure signals	Risk score; release gate recommendation
RCA Engine	Dependency graph; anomalies; lineage events	Suspected root cause; explanation; impacted assets
Policy and Remediation	SLOs; runbooks; security constraints	Auto-remediation; ticketing; rollback/backfill plans

The modular design reduces coupling and enables incremental adoption within existing toolchains. [26]

5. Methodology

The analytics engine operates on a feature store derived from telemetry and metadata. For each pipeline run or stream window, features include latency percentiles, error rates, retry counts, resource saturation signals, schema deltas, null ratios, uniqueness and duplicate rates, and distribution shift statistics (e.g., PSI, KL divergence) for key attributes. [22]

Defect prediction combines static signals (complexity, churn, dependency changes) with operational signals from recent runs (failure frequency, timeout patterns, rollback history). A lightweight model can provide a fast gate for high-risk changes, while a richer ensemble improves recall for rare but severe failures. [18]

For anomaly detection, the framework supports both unsupervised baselines (isolation forests, autoencoders) and supervised models when labels exist. Stream-based feature aggregation enables detection at multiple granularities (task, job, dataset, consumer). [28]

RCA is formulated as ranking candidate components (tasks, services, datasets) using evidence from temporal correlations, causal hints from traces, and lineage dependencies. Explanations are produced as ordered event chains with supporting telemetry slices. [27]

6. Implementation Considerations

In orchestration systems, each task emits structured start/stop events and propagates trace context across operators. Compute engines attach identifiers for stage, partition, and dataset version so that failures can be localized to specific transformations and outputs. [24]

Automated data validation is integrated as a first-class testing layer for pipelines. Validation rules may be explicit expectations (schema, bounds) or learned constraints from historical partitions. Violations can trigger quarantines, fallback to prior good partitions, or staged rollbacks. [20]

Security and resilience requirements are addressed through least-privilege collection agents, encryption-in-transit for telemetry, and policy constraints that prevent unsafe remediation actions in regulated contexts. [4]

The framework measures automation ROI using reductions in manual triage time, fewer incident tickets, improved pipeline availability, and lower compute waste from unnecessary retries and reprocessing. [2]

7. Evaluation

We define evaluation along three axes: detection (how quickly issues are identified), diagnosis (how accurately the root cause is localized), and prevention (how often defect prediction prevents incidents). The framework is assessed using historical incident logs and replayed pipeline runs to simulate production constraints. [7]

For defect prediction, we evaluate a set of ML and ensemble baselines with standard metrics (ROC-AUC, PR-AUC, F1). Across representative datasets, ensembles typically improve recall for minority defect classes while maintaining acceptable precision for release gating. [14]

For microservice-style pipeline components, we compare trace-only anomaly detection with multi-source approaches. Incorporating logs and KPIs improves both anomaly recall and localization accuracy under partial observability. [17]

Operationally, we target reductions in MTTD and MTTR and measure alert quality (precision of actionable alerts). Automated correlation reduces duplicate alerts and enables earlier detection of subtle data drift that would otherwise propagate silently. [19]

8. Discussion and Limitations

The approach depends on consistent instrumentation and on the availability of trustworthy metadata. In multi-team environments, schema conventions, naming practices, and lineage coverage can be uneven, requiring governance and incremental rollout strategies. [10]

Learning-based systems risk false positives, especially under workload seasonality and shifting business semantics. Incorporating human feedback loops and conservative remediation policies helps prevent disruptive automated actions. [5]

Cybersecurity considerations become critical when telemetry contains sensitive identifiers or customer-related signals. Observability pipelines must implement access control, retention policies, and auditability aligned with organizational security standards. [15]

9. Conclusion

This paper presented an AI-driven monitoring and observability framework for cloud data pipelines that unifies telemetry, data quality monitoring, defect prediction, and automated RCA under policy-governed remediation. By treating pipeline reliability as an end-to-end socio-technical problem, the framework supports proactive interventions that reduce incident impact and improve trust in downstream analytics and ML. [9]

Future work includes stronger causal modeling across data and service dependencies, privacy-preserving telemetry analytics, and benchmark datasets for reproducible evaluation of pipeline observability and defect prediction in real-world cloud settings. [16]

References

1. Notaro P, Cardoso F, Almeida A, Davis J, Pei D. A survey of AIOps methods for failure management. *ACM Trans Intell Syst Technol*. 2021 Nov;12(6):81.
2. Kacheru G, Bajjuru R, Arthan N. The ROI of software automation: Measuring time and cost savings. *Int J Commun Netw Inf Secur*. 2023;15(4):774–785.
3. Shankar S, Noghabi SM, Thompson JD, Arora S. Towards observability for machine learning pipelines. In: *CIDR*; 2022.
4. Pittala SK, Ashok VKC. A new era in security: Bridging information security and cybersecurity. *Int J Multidiscip Futur Dev*. 2023;4(1):69–72. doi:10.54660/IJMFD.2023.4.1.69-72.
5. Cloud Native Computing Foundation. OpenTelemetry announces support for profiling [Internet]. 2024 Mar 19. Available from: https://www.cncf.io/blog/
6. Gunda SK. Comparative analysis of machine learning models for software defect prediction. In: *Proc Int Conf Power Energy Control Transm Syst (ICPECTS)*; 2024. p. 1–6. doi:10.1109/ICPECTS62210.2024.10780167.
7. Zhang L, Sun J, Zeller A. Automated data validation: An industrial experience report. *J Syst Softw*. 2023.
8. Kacheru G. Revolutionizing healthcare: The role of artificial intelligence in clinical practice. *J Comput Anal Appl*. 2023;31(4):1546–1544. Available from: https://eudoxuspress.com/index.php/pub/article/view/3270
9. Yao Z, Su Y, Yang T, Lyu MR, Pei D. Chain-of-Event: Interpretable root cause analysis for microservices through causal event chains. In: *Proc ACM Found Softw Eng (FSE)*; 2024.
10. OpenLineage. OpenLineage API docs (OpenAPI) v2.0.2 [Internet]. 2024. Available from: https://openlineage.io/
11. Gunda SKG. The future of software development and the expanding role of ML models. *Int J Emerg Res Eng Technol*. 2023;4(2):126–129. doi:10.63282/3050-922X.IJERET-V4I2P113.
12. Foidl H, Eberlein J, Mühlbauer T. Data pipeline quality: Influencing factors, root causes of problems, and improvement measures. *J Syst Softw*. 2024.
13. Cloud Native Computing Foundation. OpenTelemetry is expanding into CI/CD observability [Internet]. 2024 Nov 4. Available from: https://www.cncf.io/blog/
14. Albattah W, Alzahrani M. Software defect prediction based on machine learning and deep learning techniques: An empirical approach. *AI*. 2024;5(4):1743–1758. doi:10.3390/ai5040086.
15. Ashok VKC. Cybersecurity for smart infrastructure and public utilities. *Int J Multidiscip Res Growth Eval*. 2023;4(2):947–949. doi:10.54660/.IJMRGE.2023.4.2.947-949.
16. Panahandeh M, Haghshenas MS, Afzal A. ServiceAnomaly: An anomaly detection approach in microservices using traces and profiling metrics. *Inf Softw Technol*. 2024.
17. Lee C, Yang T, Chen Z, Su Y, Lyu MR. Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. In: *Proc ICSE*; 2023. doi:10.1109/ICSE48619.2023.00150.
18. Gunda SK. Fault prediction unveiled: Analyzing the effectiveness of random forest, logistic regression, and KNeighbors. In: *Proc Int Conf Self Sustain Artif Intell Syst (ICSSAS)*; 2024. p. 107–113. doi:10.1109/ICSSAS64001.2024.10760620.
19. Sivva SD, Thalakanti RR, Bandari SSG, Yettapu SDR. AI-driven decision intelligence for agile software lifecycle governance: An architecture-centered framework integrating machine learning defect prediction and automated testing. 2023;4(4):167–172. Available from: https://www.ijetcsit.org/index.php/ijetcsit/article/view/554
20. Pittala SK. Cybersecurity and online safety: A critical

asset in the information era. *J Front Multidiscip Res*. 2023;4(1):576–579. doi:10.54660/.jfmr.2023.4.1.576-579.
21. Stradowski S, Biesiada A, Nowak TP. Machine learning in software defect prediction. *Inf Softw Technol*. 2023.
22. Detecting data drift and ensuring observability with machine learning automation. *J Emerg Technol Innov Res*. 2022 Aug;9(8):JETIR2208636.
23. Gregori S, Cudré-Mauroux P, Papotti P. Automating data quality validation for dynamic ingestion pipelines. In: *Proc EDBT*; 2021.
24. Red Hat Developers. Observability in 2022: Why it matters and how OpenTelemetry can help [Internet]. 2022 Apr 12. Available from: https://developers.redhat.com/
25. National Institute of Standards and Technology. AI risk management framework (AI RMF 1.0) [Internet]. 2023. Available from: https://www.nist.gov/
26. Cloud Native Computing Foundation. Jaeger v2 released: OpenTelemetry in the core [Internet]. 2024 Nov 12. Available from: https://www.cncf.io/blog/
27. A comprehensive survey on root cause analysis in microservices. *arXiv* [Preprint]. 2024.
28. Wang R, Wang J, Wang Y. Anomaly detection with a container-based stream processing framework. *Internet Things*. 2023.