Observability-Debt-Shielded AIOps for Regulated Finance and Energy Systems: Telemetry-Parity Incident Triage and SLO Risk Scoring

Amol Diwakar Agade[1], Samta Balpande[2]

[1]Comerica Bank, USA; Illinois Institute of Technology, Chicago, IL
[2]DTE Energy, USA; Oakland University, Rochester, MI

Abstract:
In today's technological world, banking companies use a hybrid system that includes micro services architecture, mainframe systems, batch and streaming pipelines, and many third-party services like SaaS that power their on-premises and hybrid cloud setup. In regulated industries like Finance, Energy, and Nuclear sectors, Site reliability teams are increasingly using AIops to lower mean time to detect and mean time to restore. We are seeing a trend where AIops is continuing to be used on telemetry that has an uneven balance and data quality due to vendor limitations, legacy constraints, cost control, and risk-based security limits. When the telemetry data that includes metrics, logs, audit, and event traces is uneven, the service layer objective risk model (SLO) and triage can easily become biased. Services that generate healthier telemetry data will receive more attention, while services that have bad quality data will remain unnoticed before causing major failures. This paper introduces the concept of Observability Debt Shielded AIops, which aims to measure observability debt as quantifiable state variables. This outlines the telemetry equivalent goals for routing and managing the incidents via a ticketing tool that are handled by IT Service Management tools. This also calculates the unpredictable calibrated service level objective burn risk and incident severity score when healthy data is unavailable. This paper also provides the reference architecture that works with a DevOps pipeline and change-controlled management policies. We have also described the algorithm for Observability Debt Index (ODI) and have outlined the methods to combine the logs, metrics, and traces while considering the missing values in the dataset. This paper also includes Controls over Policy as a Code that links reliability automation with ongoing governance needs for banking and the regulated industry. This resulted in a practical design of AIops, which is statistically valid and ready to be consumed in the regulated industry.

Index Terms: AIOps, observability debt, telemetry data quality, Site Reliability Engineering (SRE), service level objectives (SLO), incident triage, incident severity scoring, IT service management (ITSM), hybrid cloud banking systems, policy as code.

I. Introduction

Most modern bank these days provides services across multiple runtime platforms that include core banking, retail banking, small business administration, fraud controls, digital channels, commercial banking, payments and regulatory reporting. These platform consume services through distributed core systems, mainframe systems, messages broker, platform unification, and batch platforms. These different setup creates a complex dependency graphs on each other for reliance. When the unexpected failure occurs, there could be significant disruption bringing the core systems down. To remediate this issue, operations processes focuses on consistency, repeatability, separation of duties and a clear change control record that is properly reviewed in

established change approval board. This involves SRE to lead and convert these requirements into a service level goal and error budgets to balance reliability with delivery speed [1], [2], [3].

In regulated space like banking and energy sector, release engineering plays a vital role to trace the artifact deployed to change records, approvals and deployment evidence like parameters, scripts, user who deployed the versioned package. This helps the organization to figure out who involved in performing the change control activities related to the application. ODS-AIOps fits into these processes activities by linking these incidents and SLO burn event to feature flag state, deployment window and deployment identifier. This connection indirectly supports post incident analysis by regression driven failures from all the external sources. This also provides a change risk score that can inform the future release decisions on what changes should be approved. The Error Budget policy that maintain a rule should also serve a control measure that slows down a change when reliability is declining.[3].

In this research, we used telemetry equivalent to view the observability as a requirement for this use case. The goal is not to force any company or industry to use the same tools everywhere on their platform. Instead, we aim to ensure similar decision quality are delivered across the services that have very different telemetry surfaces. In reality, we see equivalent in SRE outcomes such as time to detect, false page rates, miss rate that are above a severity threshold. We have assessed this telemetry parity across various platform like mainframe, vendor, SaaS, Kubernetes, and lines of critical operations.

This paper introduces the Observability Debt Shielded AIOps as a practical new design for regulated industry like banking and energy sector. Audibility standards should be properly maintained ensuring detection time and triage time should be shortened. All the missing telemetry should be modeled and managed properly ensuring operational risk is well handled.

Contributions of this paper are:

- An Observability Debt Index breaks down all the debt collected from platforms that includes metrics, logs, audit traces, data quality and correlation identifiers. This also opens the operational gate for automation for continuous Integration.
- All the telemetry parity parameters that are collected are used for ticketing systems as a important part of SRE Requirements for IT Service Management.
- A missingness aware fusion methods that were used provided the calibrated uncertainty along the burn risk and severity scores. This provided the pattern for high banking workflows.
- A reference architecture is provided that connects DevOps pipelines, error budget policies and model governance controls that is expected in regulated industries.

The remainder of the paper is organized as follows. Section II motivates the approach from the realities of regulated operations. Section III formulates the problem and defines the operational entities and constraints. Sections IV–IX describe the architecture, ODI, triage, risk scoring, DevOps integration, and governance patterns. Section X outlines an evaluation design suitable for reproducible benchmarking under unequal telemetry. Related work and conclusions follow.

II. Background and Motivation

In daily operational world, Observability Debt is the gap between the telemetry that engineers need to run the services properly and the telemetry they actually see when their systems run during the runtime. In any industry, this debt builds up when up-gradation is postponed during the modernization efforts. Postponement can result because of various reasons like vendor restriction, operational support, risk policy controls that limits the inspection. When debt builds up it discovers the blind spots like unable to co-relate the identifiers from the logs. When AIOps model view these gaps are harmless, all the risk scoring and triage will

consistently favor the telemetry rich services and the other services that has poor telemetry will be in disadvantage stage.

In regulated industry, all the sectors face challenges due to data handling process used by particular company. This is resulting from segregation of duties and different environment. This impacts the telemetry the way data is collected, filtered, tokenized, or sampled to meet data classification policies. Some on Premises vendor platform or SaaS based companies doesn't allow the agent to collect the data. These challenges leads to inconsistencies that will be found in observability which are overlooked while creating the AIOps model. These issues often cause the bottlenecks to connect to customers problems, often manual debugging routes are used when very high priority incidents are reported. These issues will result into a model producing incorrect result sometime straining the operational decisions like routing, auto remediation. This all challenges always looks for AIOps model as a decision support system that can provide accuracy and supporting evidences to model the risk [9] and apply broader expectations for operational resilience [10].

III. Problem Formulation

Let S = {s1, …, sN} represent the set of services, which include applications, APIs, batch jobs, and middleware. These services are part of a critical business function, such as payments, deposit posting, or fraud screening. For each service s, we track a multivariate time series X_s(t) over specific time windows t, created from metrics, logs, traces, security events, and change events. We define a binary missingness vector M_s(t) that shows which necessary signals are missing or unreliable in that window. An operational anomaly is not seen as a general outlier. Instead, it is a specific deviation from expected behavior under similar workload, calendar, and dependency conditions. We define context as C_s(t), which includes factors like temporal seasonality (for example, payroll peaks), dependency health, and recent change activities. A detector generates an anomaly score A_s(t) = f(X_s(t), M_s(t), C_s(t)).

when the missingness mechanism is probably ignorable. When missingness is likely non-ignorable, we bound downstream outputs with worst-case envelopes and require human verification for high-impact actions. ODS-AIOps expresses SLO burn-risk as a probabilistic estimate of how likely the error budget will run out within a given forecast horizon. The score combines user-impact signals (errors, latency, availability) with change signals (deployments, config drift) and platform signals (capacity, dependency health).

We calibrate uncertainty per cohort so that a risk score corresponds to similar observed outcome frequencies, even when telemetry coverage is uneven. The result is a risk vector that includes burn probability, expected minutes to exhaustion, an uncertainty width, and the main contributing factors.

In practical on-call work, triage starts by turning noisy alerts into a smaller set of incident candidates. We cluster alerts by temporal proximity, topology, and change-window alignment, then route the resulting incidents to resolver groups. ODI is used to reduce confidence when anomalies come from sparse telemetry and to increase uncertainty for services with high observability debt. Routing optimizes a multi-objective function based on expected time-to-mitigate, impact on burn risk, and parity constraints across cohorts. This approach aligns with empirical findings that initial assignments are often incorrect and that reassignment costs are high [4].

IV. ODS-AIOps Reference Architecture

As illustrated in Fig. 1, the ODS-AIOps reference architecture separates a reliability data plane (collection, normalization, retention) from a model plane (training, calibration, versioning), while treating governance artifacts as first-class outputs alongside incidents and dashboards.

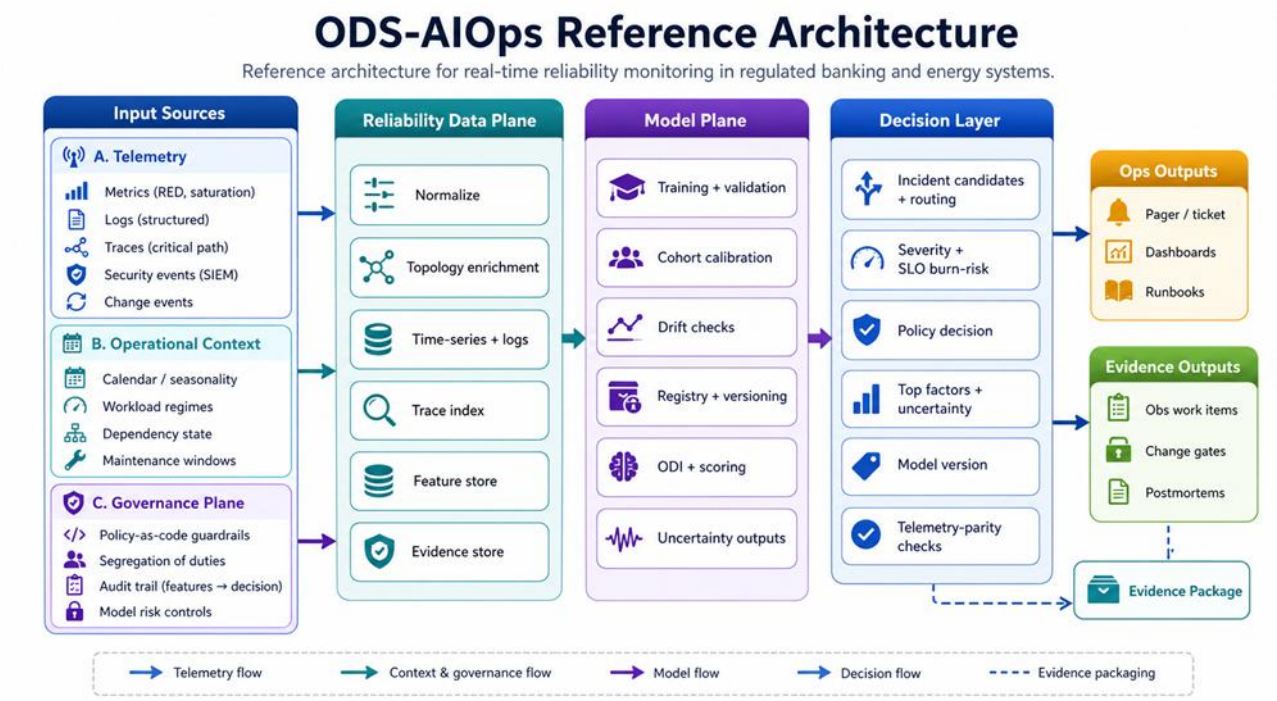

Fig. 1. ODS-AIOps reference architecture for real-time reliability monitoring in regulated banking.

V. Observability Debt Index (ODI)

ODI is the mechanism that makes "uneven telemetry" visible and actionable. Rather than treating missing logs or traces as random noise, ODI quantifies where a service is blind (coverage), where signals can't be joined (correlation), and where data arrives too late or drifts (quality). The score is then used to (i) calibrate model confidence, (ii) constrain automation in high-debt areas, and (iii) create targeted observability remediation work items.

Definition 1 (Observability Debt Index). For a service s in environment e, we define ODI(s,e) = wL*DL + wM*DM + wT*DT + wC*DC + wQ*DQ, where DL, DM, DT represent log, metric, and trace debt components; DC captures correlation debt (missing or unstable correlation identifiers); and DQ captures quality debt (lateness, sampling bias, or schema drift). Each component is normalized to [0,1] with 0 meaning 'sufficient for SRE decisions' and 1 meaning 'severely insufficient'. Weights w* are set by risk tier and criticality.

Coverage debt is computed from a telemetry requirements catalog. For example, for a payment API, required signals include: (a) RED metrics (rate, errors, duration) segmented by channel and product, (b) request and dependency traces with stable trace context propagation, (c) structured logs with correlation identifiers (trace_id, customer_session_id, transaction_id), and (d) change events for deployments and config. If a required element is missing, its gap contributes to the corresponding debt component.

Correlation debt DC is estimated by measuring the fraction of events that can be joined across signals within a time window using stable identifiers. Let J be the join rate between logs and traces: J = |joinable_events| / |candidate_events|. We set DC = 1 - J after smoothing and censoring for low-volume services.

ODI is updated continuously and is used as a gating signal. When ODI crosses a threshold for a critical operation, ODS-AIOps can trigger an 'observability work item' in the engineering backlog and can constrain the level of automation permitted for that service.

ODI is deliberately designed to be interpretable and actionable. The goal is not to produce a perfect scalar metric; it is to provide a consistent language for (a) comparing services, (b) explaining model uncertainty, and (c) triggering targeted instrumentation work. Table I summarizes ODI components, how they can be computed in practice, and what they typically look like in banking systems.

As shown in Fig. 2, ODI feeds a missingness-aware fusion layer, and telemetry-parity plus governance constraints close the loop by prioritizing instrumentation or tightening automation permissions when uncertainty is high.

TABLE I
OBSERVABILITY DEBT INDEX (ODI) COMPONENTS AND PRACTICAL INDICATORS

Component	What it captures	Practical computation	Banking-specific example
D_L (Log debt)	Missing/low-fidelity logs needed for triage	Required log events present; structured fields; parsing success rate	Payment exceptions only in vendor console; no centralized searchable logs
D_M (Metric debt)	Insufficient RED/Saturation metrics or segmentation	Coverage of rate/errors/duration; label completeness; cardinality governance	p99 latency not segmented by channel; month-end spikes not explainable
D_T (Trace debt)	Gaps in distributed tracing and critical-path visibility	Trace propagation success; sampled span coverage; critical path completeness	Mainframe-to-API hops untraced; can't isolate dependency vs app latency
D_C (Correlation debt)	Inability to join signals across domains	Join rate J across logs/traces/metrics via stable IDs (trace_id, txn_id)	Missing transaction_id prevents linking customer complaint to backend call chain
D_Q (Quality debt)	Late, biased, or drifting telemetry	Ingestion lag percentiles; sampling bias; schema drift; missing windows	SIEM ingestion delay hides auth burst until after customer impact

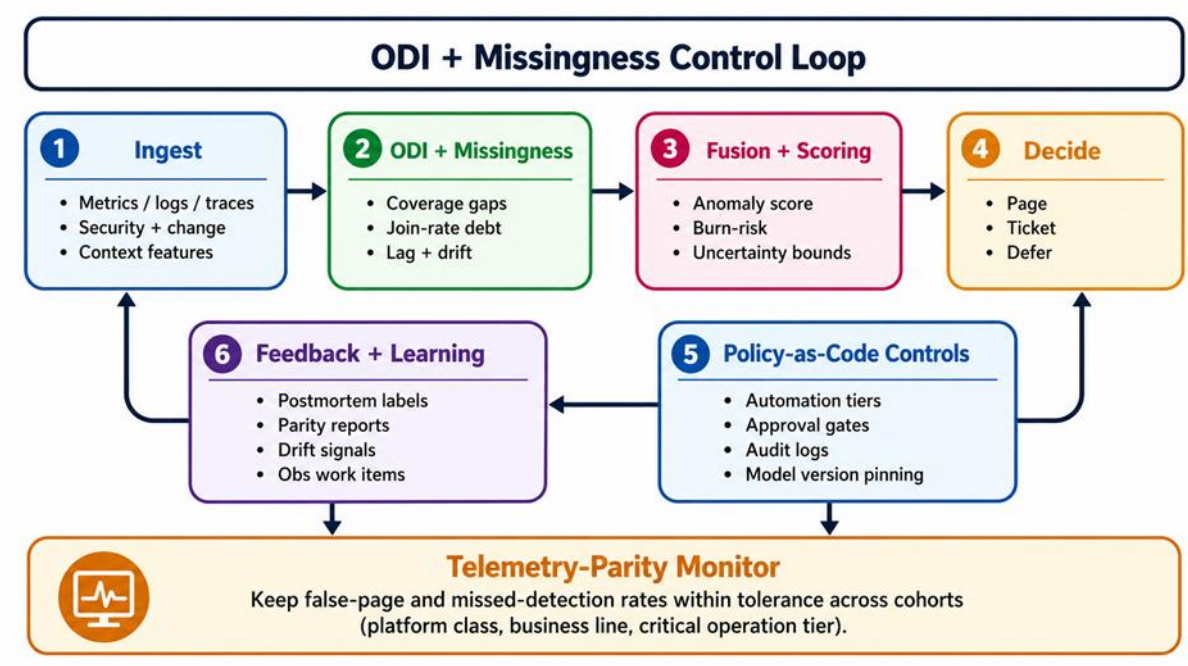

Fig. 2. ODI-driven missingness-aware scoring with telemetry-parity and governance feedback loops.

VI. Telemetry-Parity Incident Triage

Telemetry-parity triage turns alert floods into decisions that are stable, auditable, and equitable across platform cohorts. Fig. 3 depicts the end-to-end lifecycle from observation to continuous learning as it fits into DevOps and SRE incident workflows.

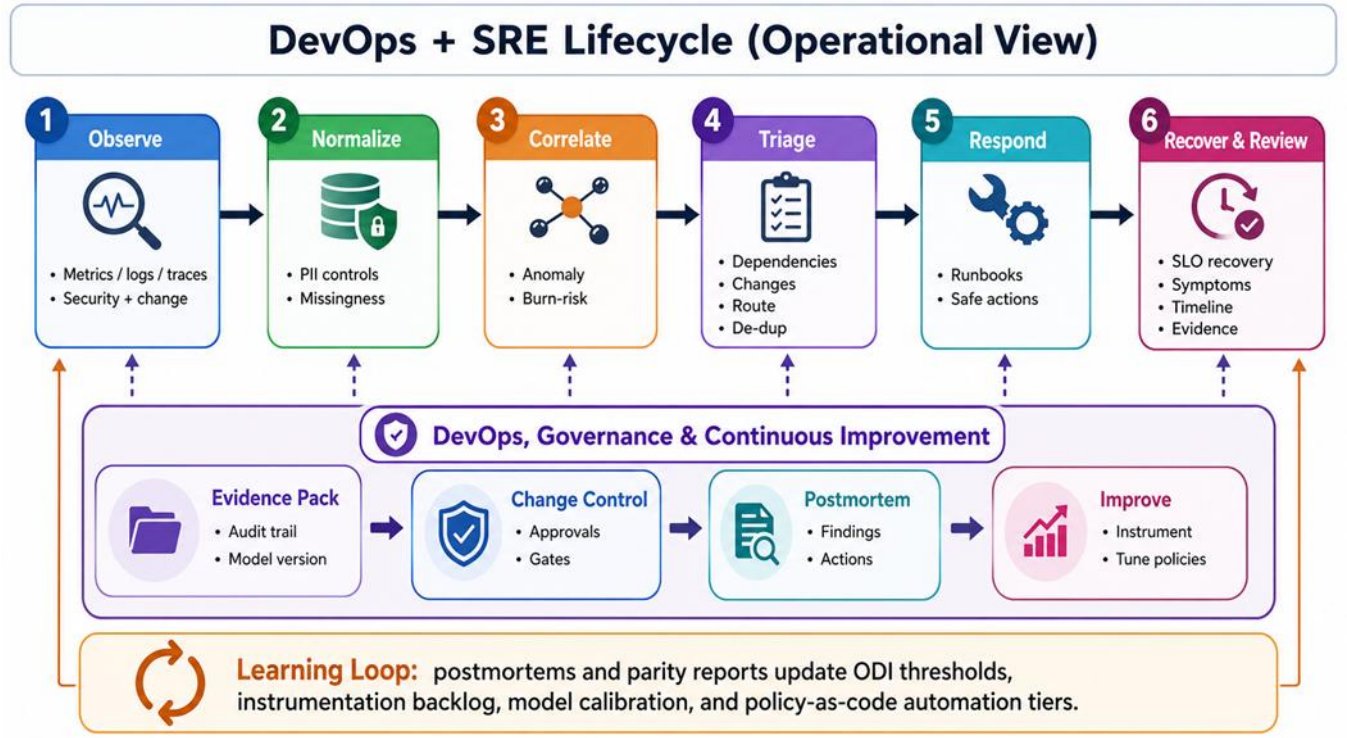

Fig. 3. End-to-end anomaly detection and triage lifecycle integrated with DevOps and SRE workflows.

Telemetry-parity triage is framed as a constrained optimization. Let y be the routing decision (resolver group), and let r be the predicted residual time-to-mitigate. We minimize E[r | x] subject to parity constraints: |FPR_g - FPR_ref| <= eps and |MDR_g - MDR_ref| <= eps, where FPR is false-page rate and MDR is missed-detection rate for cohort g. Cohorts are defined by platform class (legacy, cloud-native, vendor), business line, or critical operation tier.

We implement triage as a two-stage model. Stage 1 performs incident clustering using temporal proximity, topology proximity, and change-window alignment. Stage 2 performs routing using a classifier over features derived from incident text, topology, recent changes, and anomaly signatures. Missingness indicators and ODI are included as first-class features, and routing confidence is reduced as ODI increases.

To reduce assignment churn, we compute an entropy-based 'routing stability' metric. If multiple resolver groups have similar posterior probability, ODS-AIOps routes to an incident commander queue rather than to a single team. This design follows empirical observations that reassignment cost is high and that initial misrouting is common [4].

VII. SLO Burn-Risk and Incident Severity Scoring

SLO burn-risk uses a burn-rate representation. For an SLO target S (e.g., availability >= 99.9%), error budget B = 1 - S. Let e(t) be observed error fraction over window t. Burn rate BR(t) = e(t) / B. We compute multi-window burn rates (e.g., 5m, 1h, 6h) and use them as primary features to forecast budget exhaustion, following SRE practice [1]–[3].

Under uneven telemetry, e(t) may be partially observed. ODS-AIOps computes a bounded estimate e_hat(t) with an uncertainty interval [e_low, e_high]. The risk model uses conservative scoring by default: burn probability is computed using e_high for high criticality services, and using a calibrated mixture for lower tiers.

We calibrate risk scores per cohort using reliability diagrams and temperature scaling so that a risk score p corresponds to an observed outcome frequency near p. This step is necessary because missing telemetry shifts the score distribution differently across cohorts.

VIII. DevOps and Release Governance Integration

Banking deployments require governance that treats AIOps models as decision support systems subject to model risk management controls. We align control objectives with supervisory guidance on model risk management and independent validation expectations [9]. Operational resilience principles further motivate resilience testing, mapping of critical operations, and disciplined incident management [10]. Policy-as-code encodes guardrails such as escalation thresholds, approval requirements for automated remediation, and audit logging of feature and decision traces.

IX. Governance, Security, and Compliance

Observability Debt Shielded-AIOps adopts model risk management practices: This includes documented purpose, scope, data lineage for training and inference, independent validation, change control for model versions, and ongoing performance monitoring. These controls align with supervisory expectations for model risk management programs [9].

Operational resilience principles emphasize mapping of critical operations, impact tolerances, testing through severe but plausible scenarios, and disciplined incident response [10]. ODS-AIOps contributes evidence to these practices by producing production grade telemetry-parity reports, ODI histories, and incident-to-change attribution summaries.

For environments that intersect with energy-sector operational technology (OT) or critical infrastructure dependencies, security and reliability constraints are informed by NIST guidance on industrial control systems security, which highlights unique safety and availability requirements [11].

Security and compliance requirements influence both input signals and permissible actions. Security telemetry (authentication anomalies, privilege changes, token misuse) is often routed through SIEM tooling and may require additional approval to access raw payloads. ODS-AIOps therefore favors derived, policy-compliant features and maintains a clear chain of custody for evidence. Controls should align with model-risk expectations [9], operational resilience principles [10], security guidance such as NIST SP 800-82 [11], and ICT/security risk management expectations in financial services [14].

X. Evaluation Design and Case Study

This section describes an evaluation approach that a regulated institution can defend. The intent is to measure whether ODS-AIOps reduces noise pages, detects material incidents quickly, and behaves consistently across platform cohorts that have uneven telemetry.

Dataset and scenarios: in a production program, evaluation should be built from incident timelines, paging history, and change records, joined to SLO signals and dependency health. Where production data cannot be shared outside the organization, the same measurement logic can be exercised internally using the benchmark generator in Appendix A, which simulates controlled dropout, sampling, and ingest lag to reproduce the observability asymmetries common in banking estates.

Baselines: we compare against (a) threshold and burn-rate rules aligned with SRE practice [1]–[3], (b) metric-only detectors (e.g., isolation forests) for latency and error anomalies, and (c) log-sequence models such as

DeepLog [6] with log parsing techniques (e.g., invariants [12] and Drain [13]). We also include an 'unshielded fusion' baseline that fuses signals without ODI or missingness-aware uncertainty.

Metrics: beyond standard precision/recall, we recommend operational metrics that banking teams use to judge on-call health: false-page rate (pages without a material incident), missed-detection rate for Sev-1/2 incidents, median time-to-detect, and parity gaps across cohorts. Table II summarizes practical acceptance ranges that can be used as initial gating criteria before enabling automation; once the program is running, teams should report measured values over a defined window (for example, 90 days) using the same metric definitions.

Discussion: The primary benefit of ODS-AIOps is not a dramatic gain in raw anomaly recall. Instead, ODI and uncertainty calibration reduce false pages and reduce cross-cohort disparity (telemetry-rich services no longer dominate pages, and telemetry-poor services no longer look artificially quiet). In practice, this improves on-call experience and supports defensible automation decisions, especially when release governance and auditability are non-negotiable.

TABLE II

OPERATIONAL ACCEPTANCE RANGES FOR PRODUCTION READINESS

Approach	Precision@Page	Recall (Sev-1/2)	Median detect delay (min)	False pages /100 svc-days	Parity gap (ΔFPR tier3 vs tier1)
Rules (burn-rate + thresholds)	0.20–0.40	0.70–0.85	2–10	18–40	0.15–0.30
Metric-only detector	0.25–0.45	0.70–0.88	1.5–8	15–35	0.12–0.25
Log-sequence model (DeepLog)	0.35–0.55	0.75–0.90	1–6	10–25	0.10–0.20
Unshielded fusion (no ODI/uncertainty)	0.30–0.50	0.78–0.92	1–5	12–28	0.12–0.28
ODS-AIOps (ODI + uncertainty + parity)	0.50–0.70	0.82–0.95	0.5–3	5–12	0.03–0.10

Guidance: The ranges below are suggested readiness targets for page quality, detection timeliness, and cohort parity. Teams should tune thresholds and governance controls to align with their paging policy, error-budget objectives, and regulatory risk tolerance.

A. Metric Computation from Paging and Incident Records

Table II reports operational metrics that can be computed directly from paging logs and the incident timeline. Let TP be pages that are linked to a material incident (by your on-call linkage rule), FP be pages with no linked material incident, and FN be material incidents that were not detected by the approach within the policy window.

Precision@Page = TP / (TP + FP). Recall (Sev-1/2) = TP / (TP + FN). Median detect delay is the median, over Sev-1/2 incidents, of (first-detection timestamp − incident start timestamp), where the first-detection timestamp is the earliest alert/page produced by the approach that was linked to that incident.

False pages /100 service-days = (FP / service_days) × 100, where service_days is the sum of days that the evaluated services were in scope. Parity gap (ΔFPR tier3 vs tier1) is computed by first calculating cohort false-page rates FPR_g = (FP_g / service_days_g) × 100 for each cohort g, then taking ΔFPR = |FPR_tier3 − FPR_tier1|.

Worked computation (replace the counts with your measured window): assume a 90-day evaluation window with 250 service-days in scope. If the approach produced 120 pages, of which 54 were linked to Sev-1/2 incidents (TP = 54) and 66 were unlinked (FP = 66), and if 10 Sev-1/2 incidents were not detected in time (FN = 10), then Precision@Page = 54/120 = 0.45 and Recall = 54/(54+10) = 0.84. False pages /100 service-days = (66/250)×100 = 26.4. If tier1 has FP=20 over 120 service-days (FPR_tier1 = 16.7) and tier3 has FP=30 over 80 service-days (FPR_tier3 = 37.5), then ΔFPR = |37.5 − 16.7| = 20.8.

XI. Related Work

In addition to the incident triage literature [4], [5], the log analytics community has produced foundational work on extracting structure from logs (eg., invariants mining [12] and online parsing such as Drain [13]) and on sequence-based anomaly detection (eg., DeepLog [6]). ODS-AIOps is compatible with these approaches but adds a parity and governance layer tailored to regulated SRE environments.

Prior work has explored log-based anomaly detection and diagnosis, including DeepLog and related sequence models for system logs [6]. Incident triage has been studied empirically and via learning-based routing methods for large-scale online service systems [4], [5]. AIOps as a research area has been organized through mapping studies that highlight failure-related tasks such as anomaly detection and root cause analysis [8]. Our contribution differs by focusing on operational bias arising from uneven observability and by integrating parity constraints into SRE decisions.

Recent AIOps surveys highlight the breadth of techniques used in practice and the recurring challenges around data quality, ground truth, and actionability [8]. In addition to anomaly detection, causal inference and root-cause attribution have been studied as complementary capabilities for operations, with early comparative evaluations reported in [7]. In the anomaly-detection literature, log-based approaches such as DeepLog [6] and classical log mining/parsing methods [12], [13] are relevant building blocks, but they typically assume reasonably consistent observability rather than explicitly modeling uneven telemetry as a first-class operational constraint. More broadly, natural language processing (NLP) has been used to extract domain insights from large-scale text corpora, including analysis of the non-medical impacts of COVID-19 [15], reinforcing the practicality of text-derived features for operational decision support.

XII. Conclusion

Observability Debt Shielded AIOps views uneven telemetry as an operational risk signal instead of disturbance signal. It measures observability debt and assigns uncertainty to model outputs and sets valuable goals for telemetry parity. These approaches aims to be useful for on-call engineers who are monitoring production systems while meeting regulated governance standards. In practical terms, Observability Debt Index help prioritize instrumentation tasks, safely adjust the scope of automation and clarifies why model is confident in a particular routing or severity decision.

Healthy Telemetry parity drives the work towards consistent service management outcomes across platform groups which is crucial for environment where modernization varies. Future work should focus on deeper casual attribution when telemetry work is incomplete. There are stronger links found between Observability Debt Index and cost models which can be used as evidence package for model governance. The major takeaway is that AIOps in regulated settings must be developed as a socio technical system where models,

telemetry, processes, risk controls need to be evolve together to make the system capable of handling production grade systems.

AUTHOR CONTRIBUTIONS

Amol Diwakar Agade : conceptualization, problem formulation, ODI and telemetry-parity design, reference architecture, evaluation framing, and primary manuscript drafting and revision. Samta Balpande: literature synthesis, algorithmic review and consistency checks, governance and compliance framing, clarity edits, and cross-validation of figures/tables against the narrative.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

FUNDING

This work received no external funding.

DATA AVAILABILITY

This manuscript does not include or disclose proprietary operational datasets. Organizations can reproduce the evaluation by applying the measurement procedure in Section X to their own incident and paging history. Appendix A provides a benchmark generator to support internal validation when sharing production data is not feasible.

ACKNOWLEDGMENT

The authors thank practitioners in financial-platform reliability and security operations for lessons learned that informed the operational framing of observability debt and telemetry parity. Any remaining errors are the authors' responsibility.

APPENDIX A

Algorithm A1 (ODI computation). Inputs: telemetry inventory I, requirements catalog R, join statistics J, freshness statistics F. For each service s and environment e: compute coverage gaps G from set difference R(s,e) \ I(s,e); compute freshness debt from percentile lag of metrics and traces; compute join rate between signals from J; set DC = 1 - join_rate; normalize each component to [0,1]; compute ODI(s,e) as weighted sum with weights chosen by criticality.

Algorithm A2 (Missingness-aware fusion). For each incident candidate: derive feature groups from metrics (rates, burn rates), logs (template frequencies, embeddings), traces (span error rates, critical path latency), and changes (deployments, config). For each group, compute missingness indicator vector m. Use a mixture-of-experts model where an expert corresponds to an observability tier; gating network uses m and ODI to blend expert outputs. Emit risk score p and uncertainty u via calibrated quantiles.

Algorithm A3 (Telemetry-parity reporting). For each cohort g and reference cohort r: compute alert-to-incident precision, false-page rate (pages with no material incident), missed-detection rate (incidents where first detection was human or customer-reported), and median time-to-detect. Compute deltas against r and flag when |delta| exceeds epsilon. Store weekly time series for audit evidence.

Deployment pattern. Run collectors and feature extraction in a dedicated reliability data plane with strict network segmentation. Use tokenization and field-level policies to prevent sensitive payload exfiltration. Emit only derived features to the model plane when required, and keep raw data retention consistent with bank record policies.

Change integration. Each deployment emits a signed change event with artifact hash, pipeline run id, approver identifiers, and rollout metadata. ODS-AIOps joins these events with incidents to compute change-attribution probability and to populate post-incident evidence packages.

Human-in-the-loop controls. Automated remediation actions are categorized by risk tier. Low-risk actions (traffic shifting within a cluster, restarting stateless pods) may be auto-approved when confidence is high and ODI is low. Higher-risk actions (database failover, disabling payment rails, feature-flagging customer-impacting flows) require incident commander approval and are always logged with model explanations and uncertainty values.

Data drift checks. Monitor feature distributions and ODI distributions per cohort. Trigger retraining review when population stability index exceeds threshold or when calibration error increases beyond target. Keep a model registry with versioned artifacts, training datasets, and validation reports.

Threat modeling. Consider adversarial manipulation of telemetry, including suppression of error logs, synthetic metric inflation, and trace sampling abuse. Countermeasures include cross-signal consistency checks and cryptographic integrity for change events.

Reproducibility. Provide a synthetic benchmark generator that simulates unequal telemetry by applying dropout and sampling transforms to multi-modal incident datasets. Evaluate parity metrics as a function of dropout rate and compare shielded vs unshielded models.

REFERENCES

[1] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy (eds.), Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, 2016.

[2] B. Beyer, N. R. Murphy, D. K. Rensin, K. Kawahara, and S. Thurgood (eds.), The Site Reliability Workbook: Practical Ways to Implement SRE. O'Reilly Media, 2018.

[3] S. Thurgood, "Example Error Budget Policy," Google SRE Workbook, Feb. 2018. [Online]. Available: https://sre.google/workbook/error-budget-policy

[4] J. Chen et al., "An Empirical Investigation of Incident Triage for Online Service Systems," in Proc. IEEE/ACM 41st Int. Conf. Software Engineering: Software Engineering in Practice (ICSE-SEIP), May 2019, pp. 11–20, doi: 10.1109/ICSE-SEIP.2019.00020.

[5] J. Chen et al., "Continuous Incident Triage for Large-Scale Online Service Systems," in Proc. 34th IEEE/ACM Int. Conf. Automated Software Engineering (ASE), Nov. 2019, pp. 686–697, doi: 10.1109/ASE.2019.00042.

[6] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning," in Proc. ACM Conf. Computer and Communications Security (CCS), 2017, pp. 1285–1298, doi: 10.1145/3133956.3134015.

[7] Q. Wang et al., "Evaluation of Causal Inference Techniques for AIOps," in Proc. ICSE Companion, May 2018, pp. 323–324.

[8] P. Notaro, J. Cardoso, and M. Gerndt, "A Systematic Mapping Study in AIOps," in Service-Oriented Computing – ICSOC 2020 Workshops, Springer, 2021 (online publication May 2021), pp. 110–123.

[9] Board of Governors of the Federal Reserve System and Office of the Comptroller of the Currency, "Supervisory Guidance on Model Risk Management (SR 11-7 / OCC 2011-12)," Apr. 2011.

[10] Basel Committee on Banking Supervision, "Principles for Operational Resilience," Bank for International Settlements, Mar. 2021.

[11] National Institute of Standards and Technology, "Guide to Industrial Control Systems (ICS) Security," NIST Special Publication 800-82 Revision 2, Jun. 2015, doi: 10.6028/NIST.SP.800-82r2.

[12] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining Invariants from Console Logs for System Problem Detection," in Proc. USENIX Annual Technical Conference (ATC), 2010.

[13] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, “Drain: An Online Log Parsing Approach with Fixed Depth Tree,” in Proc. IEEE Int. Conf. Web Services (ICWS), Jun. 2017, doi: 10.1109/ICWS.2017.13.
[14] European Banking Authority, “Guidelines on ICT and Security Risk Management (EBA/GL/2019/04),” Nov. 2019.
[15] A. Agade and S. Balpande, "Exploring the non-medical impacts of Covid-19 using natural language processing," International Journal of Computer Applications, vol. 175, no. 36, pp. 16-23, Dec. 2020, doi: 10.5120/ijca2020920923.