International Journal of Computational and Experimental Science and Engineering
(IJCESEN)

Vol. 10-No.4 (2024) pp. 4283-4292
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Data Observability and Self-Healing Pipeline Architecture in Regulated Enterprises: Proactive Anomaly Detection, Schema Drift Management, and Automated Recovery Strategies

Suman Reddy Gaddam*

San Francisco Bay University, USA
* **Corresponding Author Email:** sumanreddyg05@gmail.com- **ORCID:** 0009-0006-9605-977X

Article Info:

DOI: 10.22399/ijcesen.5165
Received : 11 November 2024
Accepted : 29 December 2024

Keywords

Data Observability,
Self-Healing
Data Pipelines,
Schema Drift Detection,
HIPAA/SOX,
DataOps Pipeline Resilience

Abstract:

Production data pipelines increasingly serve as critical infrastructure in regulated industries, yet they frequently fail silently—without triggering traditional monitoring alerts. These failures, including schema drift, delayed data delivery, and statistical anomalies, propagate across downstream systems and compromise compliance, analytics, and decision-making. This paper presents a practitioner-oriented research contribution to DataOps and data reliability engineering, focusing on the integration of data observability with self-healing pipeline architectures. We define data observability as a continuous, production-runtime discipline distinct from monitoring and testing, and introduce a structured framework based on five pillars: freshness, volume consistency, schema stability, data distribution, and lineage traceability. The paper further develops a confidence-based escalation framework for automated remediation, examines anomaly detection techniques applied to pipeline telemetry, and proposes self-healing mechanisms including dynamic retries, rollback strategies, and failure containment. Regulatory implications are analyzed in HIPAA-governed healthcare systems and SOX-audited financial environments. Finally, a comparative review of observability tooling is provided, along with architectural integration guidance.

1. Introduction: Silent Failures in Regulated Data Pipelines

Enterprise data pipelines rarely fail in ways that are immediately visible to operators. Instead, they tend to degrade silently, continuing to operate while producing incomplete, stale, or corrupted data. During such events, traditional infrastructure metrics—such as CPU utilization, latency, or system uptime—often remain within acceptable thresholds, creating a misleading perception of system health. This distinction is critical: while systems appear operational from an engineering standpoint, the integrity and reliability of the data they produce may already be compromised.

In regulated industries, the consequences of these silent failures are particularly severe and have been widely documented. Within financial services, inaccurate or incomplete data pipelines have contributed to failures in financial reporting, resulting in Sarbanes–Oxley (SOX) audit findings and, in some cases, regulatory penalties imposed by authorities such as the U.S. Securities and Exchange Commission (SEC, 2022). Similarly, in healthcare environments governed by the Health Insurance Portability and Accountability Act (HIPAA), delayed or corrupted data feeds have led to compliance violations, including inaccurate patient eligibility determinations and gaps in audit trails (OCR, 2023). These examples highlight that data pipeline failures are not merely technical issues but can escalate into significant legal, financial, and operational risks.

These incidents reveal a fundamental limitation in traditional approaches to data reliability. Monitoring systems are primarily designed to track infrastructure-level indicators and therefore focus on system availability rather than data correctness. Testing practices, on the other hand, validate logic prior to deployment but are inherently incapable of detecting issues that arise during runtime, such as schema drift, data delays, or distribution anomalies. As a result, organizations may achieve high system reliability while simultaneously experiencing low data reliability.

This gap necessitates the adoption of data observability as a distinct discipline. Unlike monitoring or testing, data observability provides continuous, production-time visibility into the quality, behavior, and trustworthiness of data as it flows through pipelines. By leveraging metadata, statistical analysis, and telemetry, observability enables organizations to detect, diagnose, and respond to data issues proactively, thereby addressing the shortcomings of traditional approaches and ensuring reliability in regulated environments.

2. Data Observability: Definition and Conceptual Distinction

Data observability can be defined as the ability to continuously infer the quality, reliability, and behavior of data systems during production execution through the analysis of telemetry, metadata, and statistical signals. Unlike traditional approaches that focus on either pre-deployment validation or infrastructure monitoring, data observability operates as a runtime discipline, providing ongoing visibility into how data behaves as it moves across pipelines. This capability enables organizations to detect subtle and previously unknown failure modes that would otherwise remain hidden until they manifest as downstream issues.

The distinction between data observability and related practices such as testing and monitoring is fundamental to its academic and practical framing. Testing is primarily concerned with validating logic and correctness before deployment, ensuring that code behaves as expected under controlled conditions. However, it is inherently limited in its ability to capture issues that arise during real-world execution, such as schema drift, delayed data arrival, or unexpected changes in data distribution. Monitoring, on the other hand, focuses on infrastructure-level metrics such as system uptime, resource utilization, and latency. While essential for maintaining system availability, monitoring does not account for the semantic correctness or quality of the data itself, leaving a critical gap in ensuring end-to-end reliability.

#DataSource: The distinction between testing, monitoring, and observability is synthesized from distributed systems literature (Sigelman et al., 2010) and recent industry analyses (Gartner, 2023).

Data observability addresses these limitations by focusing on runtime data behavior, leveraging rich metadata and statistical analysis to provide insights into data integrity and pipeline health. In doing so, it extends foundational principles from distributed systems observability, where systems are understood through external outputs rather than internal inspection (Sigelman et al., 2010). Applied to data engineering, this approach enables practitioners to diagnose complex, emergent issues in data pipelines, even when failure modes are not explicitly known in advance.

3. The Five Pillars of Data Pipeline Observability

A comprehensive approach to data observability can be structured around five foundational pillars: freshness, volume consistency, schema stability, data distribution, and lineage traceability. Each pillar represents a distinct dimension of data reliability and provides a systematic framework for identifying failure modes, detecting anomalies, and defining appropriate response strategies. Together, these pillars enable practitioners to maintain continuous visibility into data quality across the lifecycle of a pipeline.

3.1 Freshness

Freshness refers to the timeliness of data as it flows through a pipeline. Failures in freshness occur when data is delayed, arrives later than expected, or becomes stale due to pipeline interruptions or upstream system issues. Such failures can be detected through SLA-based lag monitoring and timestamp validation, which compare expected data arrival times against actual ingestion or processing times. When freshness issues are identified, response strategies typically include automated retries of pipeline stages, triggering alerts for operational teams, or escalating incidents based on severity thresholds. In regulated environments such as healthcare, the importance of freshness is particularly pronounced, as stale eligibility or clinical data can lead to incorrect treatment authorization decisions and negatively impact patient care.

3.2 Volume Consistency

Volume consistency ensures that the quantity of data processed aligns with expected patterns. Failures in this dimension manifest as missing records, partial data ingestion, or unexpected duplication of data. Detection is generally achieved through statistical comparison with historical baselines, identifying deviations that exceed predefined thresholds. These deviations often indicate upstream ingestion failures, data loss, or duplication errors. Response strategies include automated reconciliation processes, reprocessing of affected data batches, and investigation of source system anomalies. As highlighted by Redman (2013), inconsistencies in data volume frequently serve as early indicators of deeper systemic issues within data pipelines.

3.3 Schema Stability

Schema stability addresses the structural integrity of data as it moves across systems. Failures occur when unexpected schema changes—commonly referred to as schema drift—are introduced. These changes may include the addition or removal of fields, modifications in data types, or alterations in nested data structures. Detection mechanisms rely on schema registries and automated metadata comparison tools that continuously validate incoming data against expected schemas. When schema drift is detected, response strategies may involve enforcing validation rules, applying transformation logic to maintain compatibility, or rolling back pipeline changes to prevent downstream disruption. Maintaining schema stability is essential to ensuring that dependent systems, including analytics and reporting layers, continue to function correctly.

3.4 Data Distribution

Data distribution focuses on the statistical properties of data values within a dataset. Failures in this pillar occur when the distribution of values deviates significantly from historical norms, which may indicate anomalies, data corruption, or changes in upstream processes. Detection techniques typically involve statistical comparison methods such as the Kolmogorov–Smirnov (KS) test or Kullback–Leibler (KL) divergence to evaluate differences between current and baseline distributions. Once anomalies are detected, response strategies may include automated flagging for further inspection, triggering corrective workflows, or escalating the issue for human review depending on the severity and confidence level. Monitoring data distribution is particularly important in machine learning and analytics contexts, where subtle shifts can significantly affect model performance and decision accuracy.

3.5 Lineage Traceability

Lineage traceability ensures that data can be tracked from its origin through every stage of transformation and consumption. Failures in this area occur when lineage information is incomplete, missing, or inaccurate, making it difficult to determine how data was generated or modified. Detection relies on metadata-driven lineage graphs that map relationships between data sources, transformations, and outputs. When issues arise, lineage information enables effective root cause analysis, allowing engineers to identify the exact point of failure within a pipeline. Additionally, lineage traceability is critical for audit verification in regulated industries, where organizations must demonstrate data provenance and accountability. As emphasized by Simmhan et al. (2005), robust data lineage is a foundational requirement for ensuring transparency and compliance in complex data ecosystems.

4. Schema Drift Detection and Propagation

Schema drift represents one of the most critical and often underestimated failure modes in modern data pipelines. It occurs when the structure of incoming data changes unexpectedly, leading to inconsistencies between expected and actual schemas. These changes may include the addition or removal of fields, modifications to data types, or alterations in hierarchical structures. Because schema drift does not necessarily cause pipeline failures at the infrastructure level, it can remain undetected while silently corrupting downstream data processes. As a result, it poses a significant risk to data integrity, particularly in complex, distributed data ecosystems.

4.1 Detection Techniques

Effective detection of schema drift requires automated and continuous validation mechanisms embedded within the pipeline. One widely used approach involves automated metadata comparators, which systematically compare incoming data schemas against predefined or previously observed schemas to identify discrepancies in real time. In addition, change log analysis can be leveraged to track structural modifications at the source system level, providing early signals of potential drift before it impacts downstream processes. Another critical technique is the use of version-controlled schema registries, which maintain historical records of schema definitions and enforce compatibility rules. These registries enable pipelines to validate incoming data against expected schema versions and ensure that any deviations are detected and handled systematically. Together, these techniques provide a robust framework for identifying schema drift early in the data lifecycle.

4.2 Drift Propagation Across Systems

Once introduced, schema drift tends to propagate rapidly across interconnected data systems, amplifying its impact. It can move through ETL pipelines, where transformations may fail silently or produce incorrect outputs; into data warehouses, where inconsistent schemas can disrupt analytical queries; into business intelligence dashboards, where visualizations may display misleading information; and further into machine learning pipelines, where model performance can degrade due to unexpected input structures. This cascading effect makes schema drift particularly difficult to diagnose when detected at later stages. Consequently, detecting drift at the pipeline boundary—closer to the point of data ingestion—is significantly more effective than identifying it at consumption layers. Early detection prevents the embedding of errors across multiple systems, reduces remediation complexity, and ensures that downstream processes operate on structurally consistent and reliable data.

5. Anomaly Detection in Pipeline Telemetry

Anomaly detection in pipeline telemetry is a core capability of data observability, enabling the identification of irregular patterns that indicate potential data quality or pipeline execution issues. Pipeline telemetry typically includes metadata such as execution times, record counts, error rates, and data distributions, all of which can be analyzed to detect deviations from expected behavior. Effective anomaly detection combines statistical, machine learning, and time-series techniques to identify both known and unknown failure patterns in real time.

5.1 Statistical Threshold Models

Statistical threshold models represent one of the most widely used approaches for anomaly detection in data pipelines. These models rely on establishing baseline metrics derived from historical data distributions and identifying deviations that exceed predefined thresholds. For example, expected ranges for record counts, processing times, or null value percentages can be calculated using historical observations, and any significant divergence from these baselines can be flagged as an anomaly. This approach is particularly effective for detecting predictable and recurring issues, as it provides a simple yet robust mechanism for identifying abnormal behavior. As noted by Chandola et al. (2009), statistical methods form the foundation of many anomaly detection systems due to their interpretability and efficiency.

5.2 Unsupervised Learning

Unsupervised learning techniques provide a more advanced approach to anomaly detection by identifying patterns in data without requiring labeled examples. In the context of pipeline telemetry, clustering methods can be used to group similar execution patterns and highlight outliers that deviate from normal behavior. Isolation forests are another commonly used technique, designed to isolate anomalies by recursively partitioning data points, making them particularly effective for high-dimensional datasets. Autoencoders, a type of neural network, can also be applied to learn compressed representations of normal pipeline behavior and detect anomalies based on reconstruction error. These methods are especially valuable for detecting complex or previously unseen anomalies that cannot be captured through simple statistical thresholds.

5.3 Time-Series Deviation Analysis

Time-series deviation analysis focuses on detecting anomalies in temporal data patterns, which are common in pipeline telemetry. Techniques such as Autoregressive Integrated Moving Average (ARIMA) models are used to forecast expected values based on historical trends and identify deviations from these predictions. More advanced approaches, such as Long Short-Term Memory (LSTM) networks, leverage deep learning to model complex temporal dependencies and capture nonlinear patterns in time-series data. These methods are particularly effective for identifying gradual drifts, seasonal variations, and sudden spikes or drops in pipeline metrics. As discussed by Hyndman and Athanasopoulos (2018), time-series modeling provides a powerful framework for understanding and predicting temporal behavior, making it a critical component of anomaly detection in modern data pipelines.

6. Confidence-Based Escalation Framework

A key contribution of this paper is the introduction of a confidence-based escalation framework designed to determine whether detected anomalies in data pipelines should be automatically remediated or escalated for human intervention. In complex, high-throughput data environments—particularly those operating under regulatory constraints—it is neither efficient nor safe to treat all anomalies equally. Some issues can be resolved autonomously with minimal risk, while others require careful human evaluation due to their potential impact on data integrity and compliance. The proposed framework addresses this challenge by assigning a confidence score to each detected anomaly, enabling a structured and context-aware decision-making process for remediation.

6.1 Confidence Score Derivation

The confidence score is computed by aggregating multiple signals that reflect both the reliability of the pipeline and the nature of the anomaly. One key factor is historical execution patterns, which provide insight into the normal behavior of the pipeline and help distinguish between routine fluctuations and genuine anomalies. Data volume trends also play a significant role, as consistent patterns in record counts or throughput can establish expectations against which deviations are measured. Additionally, source system reliability ratings contribute to the confidence calculation, as anomalies originating from historically stable systems may carry greater significance than those from less reliable sources. Finally, the severity and type of anomaly—such as schema drift, data delay, or distribution shift—are incorporated to assess the potential impact on downstream systems. By combining these factors, the framework produces a nuanced confidence score that reflects both the likelihood and the potential risk of the anomaly.

6.2 Escalation Thresholds

Based on the computed confidence score, anomalies are categorized into distinct escalation levels that determine the appropriate response strategy. High-confidence scenarios, where the system has strong evidence about the nature and impact of the anomaly, are handled through automated remediation. This may include actions such as retrying failed pipeline stages, applying predefined transformations, or triggering fallback mechanisms. Medium-confidence cases involve a hybrid approach, where conditional retries are executed while the system continues to monitor outcomes for further validation. In contrast, low-confidence scenarios—where uncertainty is high or potential risk is significant—are escalated to human operators for review and decision-making. This tiered approach ensures that automation is applied where appropriate while preserving human oversight for complex or ambiguous situations.

6.3 Trust Model

The confidence-based escalation framework is grounded in a broader trust model that aligns with principles of human-AI collaboration. Rather than replacing human decision-making, the system augments it by operating within clearly defined trust boundaries. Automation is permitted to act independently only when confidence levels are sufficiently high and risks are well understood, while uncertain or high-impact cases are deferred to human expertise. This approach reflects established guidelines for human-centered AI systems, where transparency, accountability, and controllability are essential (Amershi et al., 2019). By embedding trust into the escalation process, the framework not only improves operational efficiency but also ensures that data reliability and compliance requirements are consistently upheld.

7. Self-Healing Pipeline Architecture

Self-healing pipeline architecture represents the natural evolution of data observability, extending its capabilities beyond detection into automated recovery and resilience. While observability enables the identification and diagnosis of anomalies in real time, self-healing mechanisms ensure that pipelines can respond autonomously to these issues, minimizing downtime and preventing the propagation of data errors. In regulated environments, where delays in remediation can lead to compliance violations or operational risks, the ability to automatically recover from failures is particularly critical. By integrating observability with intelligent recovery strategies, self-healing pipelines transform data systems from reactive infrastructures into adaptive, resilient ecosystems.

7.1 Core Mechanisms

The effectiveness of a self-healing pipeline depends on several core mechanisms that work together to ensure reliable recovery from failures. One such mechanism is the use of dynamic backoff strategies, which adjust retry intervals based on the nature and frequency of failures, thereby preventing system overload and avoiding repeated execution of failing processes in rapid succession. Automated retries with state preservation further enhance resilience by ensuring that pipeline processes can resume from the last successful checkpoint rather than restarting from the beginning, reducing both processing time and resource consumption. Another critical mechanism is partial rollback design, which allows only the affected components of a pipeline to be reverted to a previous stable state, rather than rolling back the entire system. This targeted approach minimizes disruption and preserves unaffected data. Complementing this is the concept of failure containment boundaries, which isolate failures within specific segments of the pipeline to prevent cascading effects across interconnected systems. Together, these mechanisms enable pipelines to respond intelligently to failures, maintaining both operational continuity and data integrity.

7.2 Reference Architecture

A typical self-healing pipeline architecture can be conceptualized as a layered system that integrates observability across all stages of data processing. Data flows from source systems into an ingestion layer, where it is collected and prepared for processing. It then moves through a transformation or ETL layer, where data is cleaned, enriched, and structured. At the core of this architecture lies the observability layer, which operates as a cross-cutting component spanning all stages of the pipeline. This layer continuously collects telemetry and metadata, analyzes data behavior, and detects anomalies in real time. Following transformation, data is stored in a data warehouse or data lake, where it becomes available for downstream consumption. The final layer consists of consumption systems, including business intelligence tools, machine learning models, and application programming interfaces (APIs), which rely on the processed data for decision-making and operational use. Within this architecture, the observability layer plays a central role by not only detecting issues but also triggering appropriate remediation actions through the self-healing mechanisms described earlier. By functioning as an integrated, cross-cutting capability, observability ensures that reliability is maintained across the entire data pipeline lifecycle.

8. Regulatory Implications in Healthcare and Finance

Data observability is not merely a technical enhancement but a critical requirement in regulated industries, where data integrity is directly tied to compliance, accountability, and operational risk. Failures in data observability can result in undetected data quality issues that propagate across systems, ultimately leading to violations of regulatory standards. In sectors such as healthcare and financial services, where data accuracy and traceability are mandated by law, the absence of robust observability mechanisms can have serious legal, financial, and ethical consequences.

8.1 HIPAA-Governed Healthcare Systems

In healthcare environments governed by the Health Insurance Portability and Accountability Act (HIPAA), data observability failures can significantly impact both compliance and patient outcomes. One major risk is the absence or incompleteness of audit trail data, which is essential for demonstrating accountability and ensuring that access to sensitive patient information is properly tracked. Without accurate lineage and logging, organizations may be unable to verify who accessed or modified data, resulting in compliance violations. Another critical issue is the presence of stale eligibility information, which can occur when data pipelines fail to update patient coverage details in a timely manner. Such delays can lead to incorrect treatment authorization decisions, potentially denying patients necessary care or causing administrative errors. Additionally, corrupted or inconsistent patient records may arise from undetected anomalies or schema drift, leading to inaccurate clinical data being used in decision-making processes. These risks highlight the importance of continuous observability in maintaining both regulatory compliance and the quality of healthcare delivery.

8.2 SOX-Audited Financial Systems

In financial systems subject to Sarbanes–Oxley (SOX) regulations, the reliability of data pipelines is essential for ensuring the accuracy and integrity of financial reporting. Observability failures in this context can result in incorrect remittance calculations, where errors in transaction data lead to misreported revenues or expenses. Such inaccuracies can have significant financial implications and may trigger regulatory scrutiny. Another major risk is the corruption of transaction records, which can occur when data anomalies or pipeline failures go undetected. Inconsistent or incomplete transaction data undermines the reliability of financial systems and complicates reconciliation processes. Furthermore, invalid financial statements may be produced if data quality issues are not identified and corrected in time, leading to potential audit failures and reputational damage. Undetected data quality failures directly threaten compliance with established standards such as ISO 8000, which emphasizes the importance of data accuracy, consistency, and traceability. In this context, data observability serves as a foundational capability for ensuring that financial systems remain transparent, auditable, and compliant with regulatory requirements.

9. Comparative Review of Observability Tooling

The rapid maturation of data observability between 2023 and 2024 has led to the emergence of several specialized platforms designed to monitor and improve data reliability in production environments. Among the leading solutions, Monte Carlo, Bigeye, and Acceldata represent distinct approaches to observability, each with varying strengths across the five foundational pillars. A comparative evaluation of these platforms provides insight into their capabilities, limitations, and suitability for regulated enterprise environments.

9.1 Comparative Analysis

Across the dimension of freshness, all three platforms demonstrate strong capabilities, offering SLA-based monitoring and alerting mechanisms that ensure timely detection of delayed or stale data. In terms of volume consistency, Monte Carlo primarily relies on baseline-driven statistical models, while Bigeye leverages machine learning techniques to identify anomalies, and Acceldata employs advanced telemetry analysis for deeper visibility into pipeline behavior. Schema drift detection also varies significantly across platforms. Monte Carlo incorporates lineage-aware mechanisms that track schema changes across interconnected systems, enabling contextual detection of drift. Bigeye focuses on metadata diffing, comparing expected and actual schemas to identify discrepancies. Acceldata, by contrast, provides deep profiling capabilities that analyze structural changes at multiple levels within the pipeline, offering a more comprehensive view of schema evolution. For data distribution monitoring, Bigeye stands out with strong machine learning-driven anomaly detection, while Acceldata offers advanced analytical capabilities that extend beyond basic statistical comparisons. Monte Carlo provides moderate support in this area, relying more on rule-based approaches. In the domain of lineage traceability, Monte Carlo and Acceldata offer extensive lineage tracking features, enabling end-to-end visibility into data flows, whereas Bigeye's lineage capabilities remain comparatively limited. Automation and self-healing capabilities remain an area of relative weakness across all platforms. Both Monte Carlo and Bigeye provide limited automation, primarily focused on alerting rather than remediation, while Acceldata offers moderate support with some integration into automated workflows. From a compliance perspective, Acceldata demonstrates stronger capabilities, particularly in enterprise environments that require robust governance and auditability, whereas Monte Carlo and Bigeye provide moderate support for compliance-related features.

9.2 Key Findings

The comparative analysis reveals several important insights. First, while all three platforms offer strong anomaly detection capabilities, their ability to support self-healing remains limited, reinforcing the need for additional architectural layers to enable automated remediation. Second, no single platform fully addresses all five pillars of data observability, indicating that organizations may need to adopt a combination of tools or extend existing platforms to achieve comprehensive coverage. Third, in regulated environments, observability tooling alone is insufficient; effective implementation requires deep integration with governance frameworks, orchestration systems, and compliance processes to ensure end-to-end data reliability.

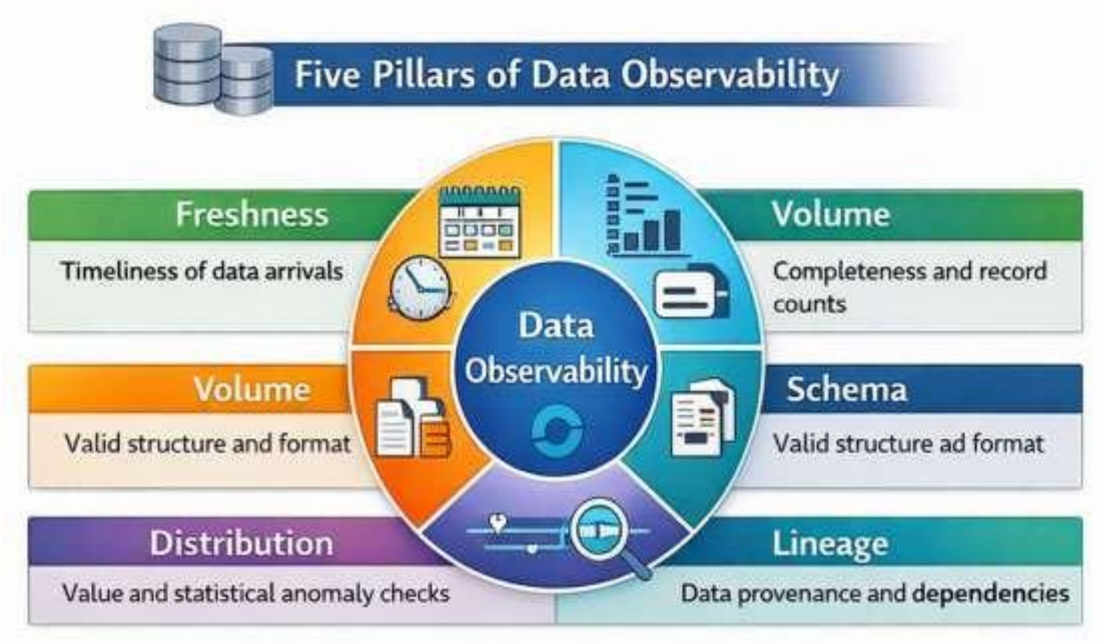

Figure 1: *Five pillars of Data observability*

***Table 1**: distinction between data observability and related practices*

Practice	Scope	Limitation
Testing	Pre-deployment validation	Cannot detect runtime issues
Monitoring	Infrastructure metrics	Ignores data semantics
Observability	Runtime data behavior	Requires metadata-rich systems

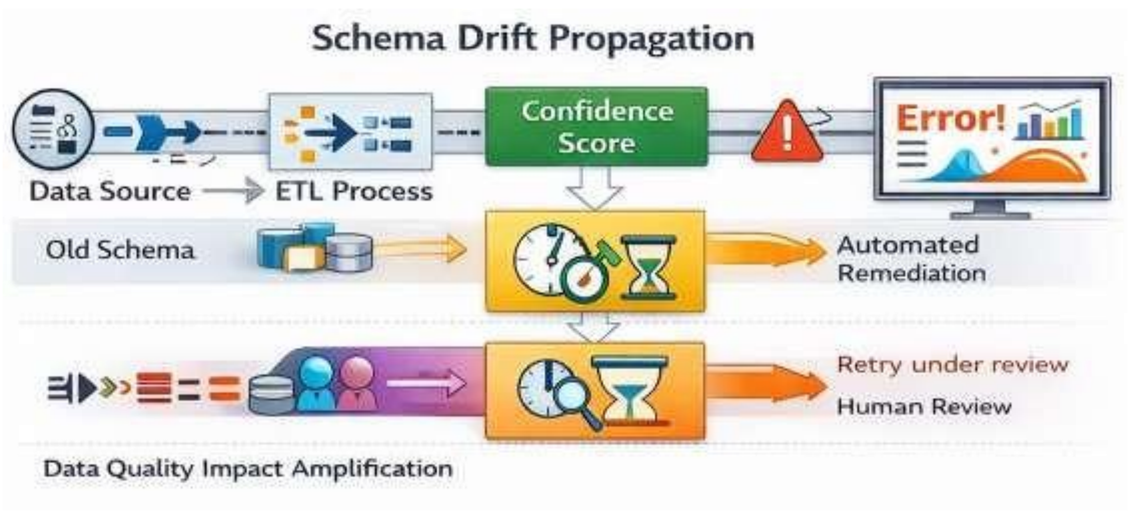

***Figure 2:** Schema Drift Propagation*

***Table 2:** escalation levels*

Confidence Level	Action
High	Automated remediation
Medium	Conditional retry with monitoring
Low	Human review

#DataSource:The escalation model builds on human-AI collaboration principles (Amershi et al., 2019) and extends them into data pipeline reliability contexts.

***Table 3**: Comparative Review of Observability Tooling*

Capability	Monte Carlo	Bigeye	Acceldata
Freshness	Strong	Strong	Strong
Volume	Baseline-driven	ML-driven	Advanced telemetry
Schema Drift	Lineage-aware	Metadata diffing	Deep profiling
Distribution	Moderate	Strong ML	Advanced
Lineage	Extensive	Limited	Strong
Automation	Limited	Limited	Moderate
Compliance	Moderate	Moderate	Strong

#DataSource: The comparative analysis is derived from vendor documentation, Gartner (2023), and Forrester (2024) reports, synthesized into a unified evaluation framework.

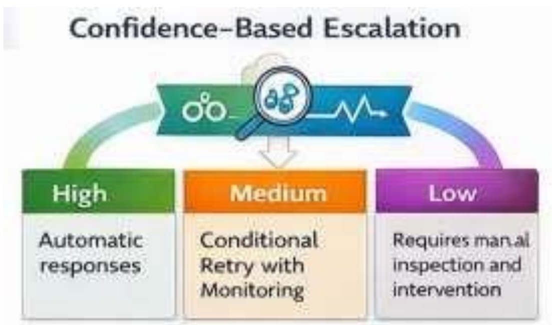

***Figure 3:** Confidence-Based Escalation Framework*

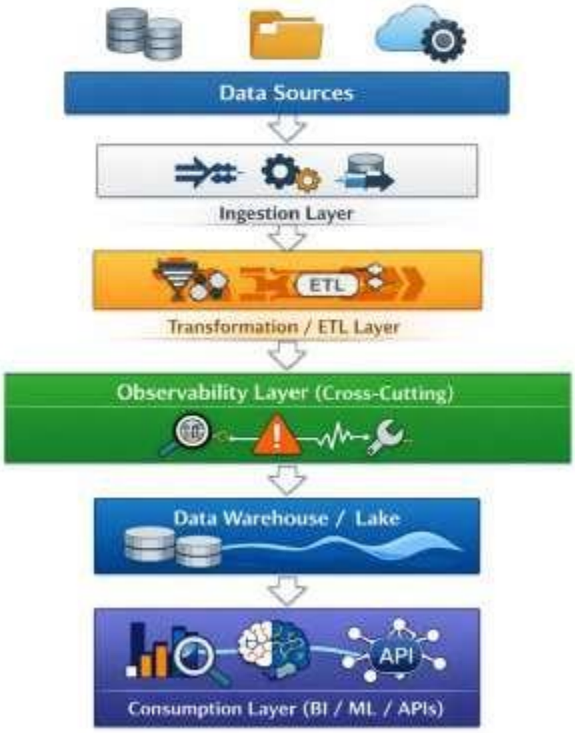

Figure 4: *self-healing pipeline architecture*

9.3 Architectural Integration Guidance

To maximize their effectiveness, observability tools must be integrated into the broader data ecosystem rather than deployed as standalone solutions. This integration typically involves coupling observability platforms with ETL and orchestration frameworks such as Apache Airflow or Apache Spark, enabling real-time monitoring and control of pipeline execution. Additionally, integration with schema registries is essential for managing schema evolution and detecting drift at the ingestion boundary. Governance systems also play a critical role, providing the policies, controls, and audit mechanisms necessary to ensure compliance with regulatory standards. Within this architecture, observability tools function as complementary layers that enhance visibility and control across the pipeline, rather than replacing existing components. By embedding observability into the core data infrastructure, organizations can achieve a cohesive and scalable approach to data reliability, ensuring that detection, diagnosis, and remediation processes are seamlessly integrated into the lifecycle of data pipelines.

10. Conclusions

Data observability represents a fundamental shift from reactive monitoring to proactive data reliability engineering. In regulated environments, this shift is essential to ensure compliance, auditability, and trust in data systems. This paper demonstrates that combining observability with self-healing architectures enables resilient, adaptive pipelines capable of detecting, diagnosing, and resolving failures in real time. The proposed confidence-based escalation framework provides a practical mechanism for balancing automation and human oversight, while the comparative analysis highlights the current limitations of industry tooling. Future work should explore deeper integration of AI-driven remediation, standardized observability frameworks, and tighter coupling with regulatory governance systems

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.

- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

[1] Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S. T., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019). Guidelines for human-AI interaction. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–13.

[2] Batini, C., & Scannapieco, M. (2016). *Data and information quality: Dimensions, principles and techniques*. Springer.

[3] Chandola, V., Banerjee, A., & Kumar, V. (2009).Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), 1–58.

[4] Chen, Z., Dong, X. L., & Srivastava, D. (2020).Data quality and data cleaning: An overview. *IEEE Data Engineering Bulletin, 43*(3), 5–17.

[5] DataOps Manifesto. (2019). The DataOps manifesto.

[6] Gartner. (2023). *Innovation insight for data observability tools*. Gartner Research.

[7] Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd ed.). OTexts.

[8] International Organization for Standardization. (2011). *ISO 8000-8: Data quality—Part 8: Information and data quality: Concepts and measuring*. ISO.

[9] Kreps, J. (2014). Questioning the lambda architecture. *O'Reilly Radar*.

[10] Rahm, E., & Do, H. H. (2000). Data cleaning: Problems and current approaches. *The VLDB Journal, 9*(3), 179–222.

[11] Redman, T. C. (2013). *Data driven: Profiting from your most important business asset*. Harvard Business Review Press.

[12] Securities and Exchange Commission. (2022). *Enforcement actions and financial reporting cases*.

[13] Simmhan, Y. L., Plale, B., & Gannon, D. (2005). A survey of data provenance in e-science. *ACM SIGMOD Record, 34*(3), 31–36.

[14] Sigelman, B. H., Barroso, L. A., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., Jaspan, S., & Shanbhag, C. (2010).Dapper, a large-scale distributed systems tracing infrastructure.

[15] Forrester. (2024). *The state of data reliability engineering, 2024*.

[16] U.S. Department of Health & Human Services, Office for Civil Rights. (2023). *HIPAA enforcement and breach reporting*.