International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(17s) 2026 pp. 356-374
https://doi.org/10.70917/ijcisim-2026-4745
Article

An Adaptive Rule-Driven Data Quality Model for Distributed Telemetry Systems

Manish Singh

University Of Illinois Urbana-Champaign, USA
ORCID: 0009-0004-6537-9637

Abstract: In distributed telemetry, machines produce a constant stream of signals as input to be used for observation, experimentation, prediction, and decision-making. Static validation rules have been widely applied for data quality assessments; however, they are unsuitable for a distributed telemetry scenario where data producers have different characteristics, traffic patterns change frequently, and the information in proximity to the data stream may be dependent not only on the data format but also on the business context. The ARDQM integrates behavior-based profiling, dynamic rule generation, multi-level rule execution, and composite and multi-action custom rules, as well as trust-aware dataset promotion in a 3-layer architecture. The ARDQM framework architecture has three main components. These are telemetry intake and behavioral profiling, adaptive rule generation and multi-level evaluation, and trust scoring and decision gating. The trust score is a weighted sum of the trust dimensions for the four validation challenges of structural, behavioral, semantic, and business validations. The composite rules represent the current data, the behavioral profiling outputs, and the historical context as conjunctions. A discrete-event simulation study over 90 simulated days for 50 heterogeneous producers, six degradation scenarios (producer dropout, schema drift, semantic drift, volume spike, correlated failure, and mild semantic drift), and 30 separate runs reveals that, when compared to static rules, ARDQM reduced false positive quality alerts by 86% under the volume-spike scenario (0.38 ± 0.04 to 0.05 ± 0.01) and improved semantic drift detection from 0.10 ± 0.03 to 0.78 ± 0.03 under the semantic-drift scenario. All of these differences to the static baseline are statistically meaningful ($p < 0.001$, two-sided Welch's t-test, n=30 runs per condition), with 95% CIs shown. The ablation analysis shows that composite rules drive the majority of the improvement in semantic drift detection, and that trust-based graded promotion drives the majority of the improvement in reduced false positives. Because the evaluation was done entirely in discrete-event simulation (and not in a production telemetry system), all quantitative results should be regarded as points of evidence for the method's performance in simulation; they are not validated in production. Deficiencies include cold-start performance, computational overhead, risk of rule drift, and a lack of real-world validation. This contribution is both practical and a concept, such that data quality becomes an adaptive and lifecycle-bound trust function rather than a static gate, though this is yet to be empirically confirmed on production traces.

Keywords: data quality, distributed telemetry, adaptive rules, trust scoring, behavioral profiling, semantic drift, composite rules, dataset promotion

1. INTRODUCTION AND RESEARCH GAP

Telemetry systems modeled on distributed computing technology (distributed telemetry) are being incorporated as essential components of new Digital Models (Digital Twins). Cloud Services, Connected Devices, Streaming Systems, and Operations Platforms continually generate signals from machines carrying data [1, 2] to support monitoring, experiments, forecasting, anomaly detection, service reliability evaluation and Business Reporting applications. As businesses increasingly rely on technology to deliver business value through their applications, Telemetry is no longer just a side effect of application behavior, but a key source of objective evidence used to support operational and analytical decision-making [3]. There is an increasing business need for Telemetry Platforms to produce decision-making quality (credible) data, regardless of scale, and consequently for the production of need-to-decide quality data. The consequences of neglecting data quality in production systems are well documented: hidden

technical debt accumulates when data pipelines lack systematic validation [19], and surveys of production machine learning lifecycles identify data quality as a persistent bottleneck that static validation alone cannot resolve [20].

Telemetry data quality presents active challenges, despite advances in serviceability and distributed processing [4, 5]. Most validation methods are based on static rules, such as field presence, schema validation, or static thresholds for event freshness and volume. These static-check validation methods are useful for providing a baseline level of system control, but they are too simplistic to meet the needs of distributed telemetry producers as the producers evolve and/or as product behavior changes over time, and where downstream interpretation depends on business context rather than purely structural factors.

The main challenge that has been identified is that distributed telemetry systems require a more adaptive, context-aware approach to quality than customary static validation frameworks.

Selected research contributions include:

1) An architecture for adaptive telemetry data quality based on behavioral profiling, dynamic rule synthesis, and trust-aware promotion (Section 5).

2) Trust scores are subject to dimension weights as defined in policies and promotion thresholds; see Section 7.

3) A composite and multi-action rule structure that supports conjunctive conditions across multiple signals, and graded responses based on the severity or importance of the rule being applied (Section 6)

4) The comparison of ARDQM's performance with that of static checks and profiling-based checks under six different controlled degradation conditions was conducted using a discrete-event simulation model, yielding statistically meaningful results. The discussion of results from the sensitivity analysis of key hyperparameters is presented in Section 8.

5) A limitations analysis identifying cold-start behavior, computational overhead, rule drift risk, and external validity constraints (Section 9).

6) We also compare to ADWIN online drift detection [13] in Section 8, which is the strongest available single-signal adaptive method, serving as our fourth experimental baseline.

All of the key methods in ARDQM: behavioral profiling, adaptive rule thresholds, drift detection, trust scoring, and policy/threshold-based gating are derived from previous work (see Section 3) and thus not a direct contribution. The main contribution of this work is the system architecture and governance that provide quality for distributed telemetry. To the best of our knowledge, no other system has used all of these components together: continuous behavioral profiling for rule evolution, composite multi-signal rules with partial-satisfaction scoring, governed updates with safety bounds and rollback, graded trust scoring across four validation dimensions, and policy-aware promotion augmented by critical-failure overrides. We evaluate the system in Section 8 by performing ablation studies on each component and comparing it against baselines that implement some subset of this functionality.

2. WHY STATIC FRAMEWORKS BREAK DOWN IN TELEMETRY ENVIRONMENTS

Most customary systems were built under the assumption that they would be used against relatively stable enterprise-type data sources where things like schemas, refresh intervals, and business rules change slowly and in predictable ways [4]. This is not the case in distributed telemetry environments where most of these assumptions do not hold. The producers of these events could be different on different devices, services, countries, and versions of the client software. There are also rapid fluctuations in what data is transported (inbound or outbound, or whether presented at all to users). Such fluctuations may arise from changes in the composition of an organization (e.g., launching a new product), system testing, seasonality, or outages. Optional fields may become important later, and the distribution may change due to healthy business growth or actual degradation of data integrity. In environments

with continuous monitoring and automatic detection of possible problems by means of defined rules, the rules may be too strict (e.g., generating too many 'false positives') or too lenient.

A. The Interpretive Trust Problem

Another deep problem is that classical data quality checkers do not check whether the data is trustworthy from an interpretive, i.e., not technical, perspective [6]. Even when data is conformant to the schema, problems can arise because of partial failure of data producers, late-arriving data, inconsistency of data metrics, semantic drift of data values, etc. This means that a dataset can comply with the definition of completeness but nevertheless lead to an unsafe experience with the downstream data consumption. Telemetry data quality may be considered not only in a static field validation framework, but also in the context of situations when the way telemetry has been processed is sensitive, when there are cross-signal dependencies, or when there is an increasing sensitivity to the downstream effect of a decision made using telemetry.

3. RELATED WORK

In this section, we present existing data quality tools and contrast them against ARDQM's features.

Great Expectations [8] provides support for many different types of expectations as well as pipeline integration and adding new expectations. However, expectations must always be specified by an user by hand and will not change unless the user modifies them explicitly. Additionally, as telemetry processing speeds up, the latency between changing an expectation (the time between a change in behavior and a change in the corresponding expectation) increases the overhead of managing expectations for users. There is no way of directly modifying expectations based on performance profiling or promoting an expectation based on the level of confidence.

Amazon Deequ's [9] (Schelter et al.) profiling approach is a step in this direction. However, data constraints are only created once as a snapshot and are not kept up-to-date to act as a base for delivering continuous monitoring of data quality, while the telemetry data may change throughout its life cycle. However, Deequ does not support composite multi-condition rule sets or graded trust-state classifications for downstream consumption.

Monte Carlo Data [10] is a data quality tool that detects deviations from expectations by learning a baseline of incoming data feeds and triggers alerting circuitry (circuit breakers for detection purposes). However, it focuses on behavior drift detection and has no way to enforce differentiated policy or rule set actions against downstream data.

Breck et al. [11] described a data validation framework employed by Google. The framework conducts data schema validation, recognizes structure, finds statistics, and establishes a trust-state management lifecycle utilizing learned rules.

Gama et al. [13] survey online concept-drift detectors, such as the Adaptive Windowing algorithm (ADWIN), Drift Detection Method (DDM), and Early Drift Detection Method (EDDM), which focus on a single univariate performance stream and output an alarm when a statistical distribution change is detected. These methods descend from the foundational continuous inspection scheme introduced by Page [12], which established the sequential change-point detection paradigm on which modern drift detectors are built. ADWIN is the most general of the three: it has a variable-length window that can automatically grow and shrink depending on the rate of change and does not require setting the window size in advance. DDM and EDDM assume a binomial distribution of classification errors that may not be applicable to continuous telemetry parameters. Rabanser et al. [14] generalize drift detection to multivariate streams using two-sample hypothesis tests (the current state of the art for single-signal adaptive monitoring) and report results using ADWIN as the fourth baseline (Config D) in Section 8 of their paper. However, they do not support composite multi-signal conditions, trust-state management, or policy-differentiated promotion decisions at scale as would be expected of an ARDQM.

Batini et al. [5] define the types/dimensions of data quality and some of the metrics that can be used to measure them (both subjective and objective). Pipino et al. [6] list some of the objective and subjective metrics of data quality. While Wang & Strong [7] provide the foundational data quality definition of "fitness for use" and the foundational

elements of the identification of dimension(s) of quality, to our knowledge, these articles do not discuss whether adaptive rules evolve over time and/or what rules may confer trust to streaming telemetry.

Many authors proposed approaches centering on data contracts and observability tools. Dehghani [22] proposed the concept of a data mesh. Based on the theory that organizations will be successful only to the extent that there is data ownership and product thinking in the domain, data mesh can be seen as an organizational design. At the same time, Jones [25] has defined the concept of data contracts as a way for data producers and consumers to come to agreement about schemas and semantics as well as service level agreements (SLAs). Sridharan [26] has also taken observability principles from the single-system literature and applied them to distributed systems more generally through structured telemetry and trace data.

Where there are deficits in their overall coverage of all the frameworks, according to the ARDQM, Table 1 shows how this deficiency is due to the design architecture of the ARDQM, rather than any lack of capability. However, existing tools generally have a greater advantage in terms of community support, integration into existing ecosystems, and production readiness; ARDQM has only been studied as an evaluated methodology through simulation.

Table 1. Comparison of data quality frameworks. ✓ = native support; partial = limited support; – = absent. ARDQM capabilities are simulation-validated only.

Capability	Gt. Expect	Deequ	Monte Carlo	ADWIN/DDM	ARDQM
Static rule validation	✓	✓	✓	partial	✓
Profiling-driven rule update	–	partial	partial	✓	✓
Continuous rule evolution	–	–	✓	✓	✓
Composite multi-cond. rules	partial	–	–	–	✓
Multi-action rule responses	–	–	partial	–	✓
Graded trust scoring	–	–	–	–	✓
Critical-failure override	–	–	–	–	✓
Policy-aware promotion	–	–	–	–	✓
Production ecosystem maturity	✓	✓	✓	partial	–
Community adoption	✓	✓	✓	✓	–

4. PROPOSED MODEL: THE ADAPTIVE RULE-DRIVEN DATA QUALITY MODEL

Quality controls themselves are not static. ARDQM generates, evolves, and ultimately enforces quality rules based on the output of profiling systems, associated schema metadata, historical quality baselines, producer characteristics, and downstream intent. Quality gates do not lock data into a yes or no state; they provide feedback on whether the telemetry is good enough to be used downstream. For this reason, data quality is a continuously running

process. This architecture combines five elements into a governed telemetry quality architecture: (1) continuous profiling as the empirical basis for telemetry behavior, (2) adaptive rule evolution with governed update policies, safety bounds, and automated rollback, (3) composite multi-signal rules that correlate evidence across fields, datasets, and time windows, (4) graded trust scoring based on structural, behavioral, semantic, and business validation, and (5) policy-aware promotion that is aware of the dataset quality state and the use case sensitivity of the data consumers.

5. THREE-LAYER ARCHITECTURE

A. Layer 1: Telemetry Intake and Behavioral Profiling

Telemetry from distributed producers has retained lineage, timestamps, source identities, and payloads. Profiling looks for field presence, null patterns, freshness, volume, cardinality, distribution changes, and historical trends to provide an empirically grounded basis for the downstream rule logic. This can help to distinguish normal evolution from suspicious deviations by providing a profile of expected behavior.

Formally, let the telemetry input at time window t be D_t, and let $P(D_t)$ be the profiling output vector of distributional statistics over fields $f_1, \ldots, f_k$. Thus, the profiling layer maintains a rolling baseline:

$$\underline{P_t} = \frac{1}{W}\sum_{i=t-W}^{t-1} \quad P(D_i) \tag{1}$$

over a configurable window W, against which deviations are measured. The deviation for field f_j at time t is:

$$\delta_j(t) = \frac{|P_j(D_t) - \underline{P_{j,t}}|}{\hat{\sigma}_{j,t}} \tag{2}$$

where $\hat{\sigma}_{j,t}$ is the estimate of P_j's standard deviation over the baseline time window. A field-level anomaly is triggered if $\delta_j(t) > \delta t\ \delta thresh$, where δthresh is a user-configurable sensitivity parameter.

B. Layer 2: Adaptive Rule Synthesis and Validation

Quality rules are generated or updated based on profiling, schema metadata, the history of past rules, business policies, and domain expectations. Validation is multi-faceted:

- Structural integrity: schema conformance, type correctness, required field presence.
- Behavioral continuity, including: volume, freshness, and producer coverage, relative to baseline periods.
- Cross-field and cross-dataset referential checks are logical consistency checks.
- Business meaning: the metric corresponds to definitions downstream.

The rules are updated according to a drift threshold policy. This means that for the parameter θ_r of the rule r to be updated to a new parameter value θ'_r the profiling evidence must exceed a drift threshold δ, which can be specified by the user.

$$\left|\theta'_r - \theta_r\right| > \delta_r \; and \; n_{evidence} \geq n_{min} \tag{3}$$

where δ_r is a threshold drift for a specific profiling rule, and n_{min} is the minimum number of profiling windows necessary to integrate a particular change. Rule changes are versioned, and the profiling evidence responsible for changes is stored with them so that all rule changes are audited. For high-sensitivity rules, it is possible to set up a human-in-the-loop review before automatically applying any updates.

C. Layer 3: Trust Scoring and Decision-Gating

Finally, the third layer, called "inspector," translates the results of validation to a dataset trust state and decides whether the data should be promoted, delayed, flagged, or served conditionally (Section 7).

Figure 1 shows the complete data flow through the three-layer architecture.

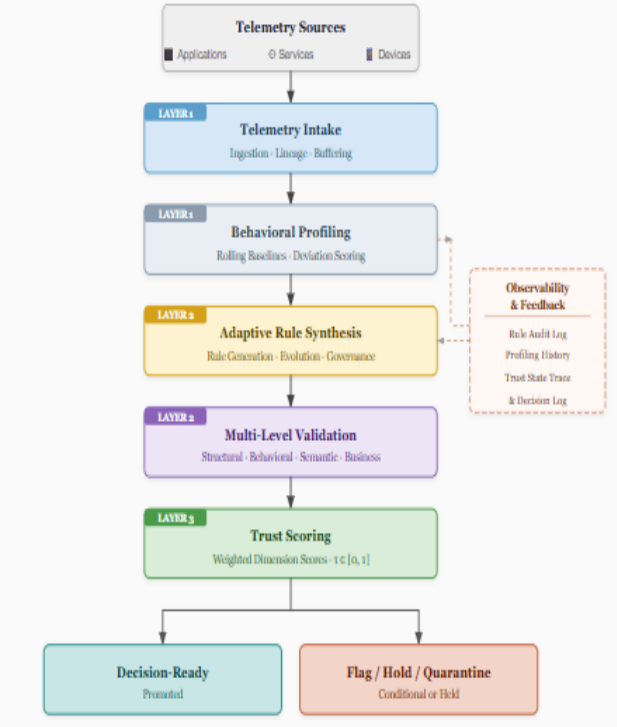

Fig. 1. Adaptive Rule-Driven Data Quality Model architecture. Telemetry flows from sources through intake, behavioral profiling, adaptive rule synthesis, multi-level validation, and trust scoring before reaching decision-ready promotion or flag/hold/quarantine paths. Dashed arrows represent observability and feedback loops that inform profiling and rule evolution.

6. COMPOSITE AND MULTI-ACTION CUSTOM RULES

Existing frameworks are usually either mainly concerned with a single quality aspect such as null checking or schema compliance [9], but quality issues often emerge when signals are combined. For example, a dataset may be structurally valid, but unsafe when combined with a freshness lag, a drop in coverage, and metric inconsistency.

A. Composite Rule Formalization

In this framework, a composite rule R^c is defined as:

$$R_c = \Lambda_{i=1}^{n} c_i(D_t, P_t, H_t) \quad (4)$$

Conditions c_i can access the dataset that they operate on D_t and context information such as produced profiling output P_t or historical context H_t (e.g. previous baseline, status of other datasets), and conditions can compare multiple datasets or time frames, multiple producer groups, and multiple historical baselines before making a decision.

The conjunction operator in Equation (4) requires all conditions to be satisfied for the composite rule to be satisfied altogether. Partial satisfaction is also a useful concept in some cases. For example, if m out of n conditions are satisfied, the ratio $\rho = m/n$ could be an additional input to the trust scoring function (see Section 7) that would grade the quality.

B. Multi-Action Rule Responses

The multi-action rule maps the composite evaluation outcome to a response set ordered by severity:

$$A = \phi(Rc, severity, policy) \in \{alert, quarantine, escalate, approve\_conditional, annotate, holdback\} \quad (5)$$

The action selection function ϕ will additionally take into consideration the composite rule outcome R^c, which includes the partial satisfaction ratio ρ. A severity classification, derived from the observed violations, and a policy context, which defines the degree of strictness of the consumers of the dataset, will also be considered. For example, when the dataset is destined for regulatory reports, the holdback may be at $\rho < 0.9$, while for exploratory analysis, the consumer may be satisfied with a holdback at the same ρ.

This is important, as most telemetry quality failures are not the result of a technical failure but of operational circumstances that must be interpreted in context instead. A multi-step classification process that allows policy-aware action paths more closely resembles how telemetry systems operate in reality, where trusting decisions are made based on multiple, coordinated pieces of evidence.

C. Rule Evolution Governance: Approval, Safety Limits, Audit, and Rollback

As Layer 2 can automatically change rule parameters (Equation 3), all changes to architecture rules have explicit governance procedures for evidence-gated approval, hard safety limits, a versioned audit trail, and mechanisms for reversal or rollback (braking).

Approval. Low- and medium-sensitivity rules may thus be updated as soon as the evidence condition in Equation (3) is satisfied. High-sensitivity rules that govern regulatory, privacy, or business-critical aspects require human-in-the-loop approval before sending the candidate update as an active update. The candidate update is held in a queue while the old parameter is used until it is approved or the queue times out.

Safety limits. Each adaptive parameter θ_r has a policy-configured hard interval $[\theta\min,r, \theta\max,r]$, and its value must not lie outside this interval, even if evidence suggests otherwise. This prevents a slow degradation from being absorbed into the rolling baseline until the rule quietly stops reporting real issues (see Section 9C).

Audit trail. Each approved, rejected, or queued change is recorded with, among other parameters, the profiling information that triggered the change, the prior and new parameter value, the automated policy or named reviewer approving the change, and a timestamp.

Rollback. Each accepted update enters a canary monitoring stage of length W_canary. If the effective value of the affected rule's detection rate or false positive rate drops below what has been configured, the change is reverted automatically, and the last known good version of the rule is applied, with logging. Operators can also force a rollback to any desired version of the rule on demand.

Algorithm 1: Governed Adaptive Rule Update

Input: rule r, current parameter θr, profiling evidence stream

Output: applied parameter θr (possibly unchanged), audit log entry

```
observe Δr(t), wr for rule r at time t
if |Δr(t)| ≥ δr and wr ≥ n_min then
    θ'r ← candidate_update(θr, Δr(t))
    θ'r ← clip(θ'r, θ_min,r, θ_max,r)        // hard safety bound
    if sensitivity(r) = high then
        queue θ'r for human approval; log("queued", r, θr, θ'r, t)
        return θr                            // unchanged pending review
    else
        apply θr ← θ'r ; log("applied", r, θr, θ'r, t)
```

```
10        start_canary_window(r, W_canary)
11 if r has an active canary window then
12     if detection_rate(r) or fp_rate(r) breaches tolerance then
13         θr ← last_known_good(r); log("rollback_auto", r, t)
14 if operator issues manual_rollback(r, version) then
15     θr ← version; log("rollback_manual", r, t)
16 return θr
```

7. TRUST SCORING AND POLICY-AWARE DATASET PROMOTION

A. Trust Score Formulation

Unlike the above, readiness is, as a single value, represented as a trust score $\tau \in [0, 1]$ which is not binary but a graded estimate of the model's readiness, computed as a weighted average of the performance results of the validation process executed on each model readiness dimension.

$$\tau(D_t) = \sum_{d \epsilon D} \quad w_d \cdot s_d(D_t) \qquad (6)$$

where D = {structural, behavioral, semantic, business} are the validation dimensions, sd ∈ [0, 1] is the normalized score for dimension d, and wd are policy-configurable weights that satisfy Σwd = 1. Dimension weights are set by platform operators to correspond to the relative importance of each quality aspect for a given dataset's use case.

Each dimension score sd is given by the average pass rate of all validation rules configured for that dimension:

$$s_d(D_t) = \frac{1}{|R_d|} \sum_{r \epsilon R_d} \quad 1|r(D_t) = pass| \qquad (7)$$

where Rd denotes the set of all rules in dimension d and 1|·| is the indicator function. For composite rules, the partial satisfaction ratio ρ is used instead.

B. Critical Failure Override

A weighted average, by construction, allows strong performance on some dimensions to offset failure on others, which is acceptable for ordinary trade-offs between competing qualities, but unacceptable for a small set of failure classes where the effects do not average away. These include undetected regulatory non-compliance, mishandling a privacy field, unreconciled cross-system totals, and a business process metric with severe semantic drift. Therefore, ARDQM defines a policy-based set of critical rules Rcrit ⊆ ∪d Rd and applies a hard override to the promotion decision, which takes precedence over the weighted score:

$$Override = 1 \; if \; \exists \, r \, \epsilon R_{crit} \; such \; that \; r \; fails, else \; 0 \qquad (8)$$

$$PromotionState = Held, whenever \; Override = 1, regardless \; of \; \tau \qquad (9)$$

In other words, a dataset cannot reach the decision-ready or conditional state solely based on a high aggregate τ on its own. Regardless of the remaining dimensions' performance, if any designated critical rule fails, the override pushes the Held state. Platform operators explicitly designate the membership of the R_crit group; this designation is versioned and audited using the same mechanism.

C. Promotion States

Datasets are classified into one of three promotion states using configurable thresholds.

- $\tau \geq \theta High$: Decision-ready – promoted for full downstream consumption.
- $\theta Low \leq \tau < \theta High$: Conditional – served with quality annotations; consumers are notified of known limitations.
- $\tau < \theta Low$: Held - quarantined pending review or remediation.

D. Policy-Aware Promotion Thresholds

To accommodate the different criticality of telemetry datasets, the architecture provides support for promotion logic where the trust thresholds (θHigh, θLow) correspond to the downstream sensitivity of the use case, as shown in Table 2. These are example default values based on thresholds recommended in SRE literature [15] and should be established via empirical ROC analysis of production deployments with historical data of quality events, taking into account the appropriate trade-offs between false-positive and false-negative rates for the particular use case.

TABLE 2. Example policy-aware trust threshold configurations by use case sensitivity. Values are illustrative defaults; production calibration is recommended.

Use Case	θHigh	θLow
Exploratory analysis	0.70	0.50
Product experimentation	0.85	0.70
Executive / business reporting	0.90	0.80
Operational alarms	0.92	0.85
Regulatory outputs	0.97	0.92

8. EVALUATION

In order to validate the claims made by the model in a lab setting, a discrete-event simulation study is conducted. All results from this section (including the main comparison and ablation and sensitivity analyzes) are based on simulations. None of the evaluations use production telemetry data. Although the simulation parameters are based on trends in distributed systems [1] and experimentation in general [15, 16], the lack of real-world validation on production systems makes it hard to generalize the quantitative results. Improvements in this way should be seen as potential architectural capabilities under ideal settings rather than guaranteed production-level improvements.

A. Simulation Design

The Python SimPy library is used to implement a discrete-event simulation of the telemetry pipeline processing a set of event streams from 50 heterogeneous producers over a period of 90 days. Each producer's event stream is modeled using a Poisson process with $\lambda = 100$ events/hour and a producer-specific variance from the distribution $N(1.0, 0.15^2)$. The profiling window length is $W = 7$ days (Equation 1), and the profiling deviation threshold at each point in time is δ_t $\delta thresh = 2.5$ standard deviations (Equation 2). These model parameters and degradations are fully specified for reproducibility. The simulation is implemented in Python 3.10 and SimPy 4.0. Random seeds for 30 replications are drawn from the set $\{1, 2, \ldots, 30\}$. The experimental design follows established guidelines for controlled experimentation in software engineering, using independent replications with fixed seeds to support statistical analysis and reproducibility [21].

Simulation parameters were derived from published observations: Poisson arrival modeling is typical for telemetry [1]; a 20% failure rate occurs in post-mortem [15]; and a 3× volume increase is standard for product launches [16].

B. Reproducibility and Code Availability

Although the simulation code was not released, all simulation parameters, random seed schedules, degradation injection schedules, and baseline configurations are provided in Table 3 to allow independent reimplementation. The discrete-event simulation uses the libraries of Python SimPy, NumPy, and SciPy to implement the degradation models in Section 8C according to the controlled injection schedules described below. This should enable an independent researcher to reconstruct the simulation environment and replicate the results as described later in this section.

Table 3. Complete simulation parameter specification for reproducibility. Implementation uses Python 3.10, SimPy 4.0, NumPy, and SciPy.

Category	**Parameter**	**Value**
General	Simulation duration	90 days
	Number of producers	50
	Event arrival process	Poisson, $\lambda = 100$ events/hour per producer
	Producer-specific variance	$N(1.0, 0.15^2)$ applied as multiplier to λ
	Number of profiled fields (k)	12
	Random seed sequence	{1, 2, …, 30}
	Runs per condition per configuration	30
Profiling (Layer 1)	Profiling window W	7 days
	Deviation threshold δthresh	2.5 standard deviations
	Baseline smoothing	Exponentially weighted moving average ($\alpha = 0.3$)
	Cold-start fallback period	W (first 7 days; static rules only)
Adaptive rules (Layer 2)	Rule drift threshold δ_r	2.0 standard deviations
	Minimum evidence windows n_{min}	3
	Hard safety bounds [θ_{min}, θ_{max}]	[0.5× initial, 2.0× initial] per parameter
	Canary monitoring window W_{canary}	3 days
	Canary rollback tolerance	≥ 10% degradation in detection rate or FP rate
Trust scoring (Layer 3)	Dimension weights w_d	Uniform: 0.25 each (structural, behavioral, semantic, business)
	Promotion threshold θ_{High}	0.90
	Promotion threshold θ_{Low}	0.70
	Critical-rule override	Enabled for semantic and business dimensions

Config A (Static)	Volume bounds	±30% of Day 1–14 mean
	Null rate threshold	< 5% per field
	Schema validation	Exact match to registered schema
	Freshness threshold	< 2× mean inter-event interval from Day 1–14
Config B (Profiling)	Profiling snapshot	Taken once at Day 14
	Constraint derivation	Mean ± 3σ from snapshot statistics per field
	No subsequent updates	Constraints fixed after snapshot
Config D (ADWIN)	ADWIN significance level α	0.01
	Application scope	Independent per profiled field
	Drift response	Flag field as anomalous when alarm fires
	No composite rules or trust scoring	Single-stream monitoring only
Degradation schedules	1. Producer dropout	Days 30–32; 20% of producers (10 of 50); duration 48 hours
	2. Schema drift	Days 45–52; 1 high-cardinality field; required → optional; rolling across producers
	3. Semantic drift	Day 60; session_count aggregation 24h → 48h (100% change)
	4. Volume spike	Days 75–78; 3× baseline volume across all producers
	5. Correlated failure	Days 82–85; 15% producer dropout + 2× freshness latency + partial schema inconsistency across affected group
	6. Mild semantic drift	Day 88; session_count aggregation 24h → 28h (17% change)

C. Degradation Conditions

The simulation contains six degradation scenarios; the first four are injected independently at regular intervals, and the last two (correlated failure and mild semantic drift) have a longer duration to probe boundary conditions:

1) Producer dropout (Days 30-32): 20% of producers drop out and stop emitting events for 48 hours.

2) Schema drift (Days 45-52): A high-cardinality field changes its status (from required to optional) during a rolling producer upgrade.

3) Semantic drift (Day 60): Although the schema is the same (engagement metric: session_count), the output of that schema has semantic drift in that it is counted over a 48-hour rather than a 24-hour time window.

4) Volume spike (Days 75-78): A spike of up to 3 times normal event volume is seen for a new product.

5) Correlated failure (Days 82-85): A simulated network partition kills 15% of the producers, with late arrivals (2x normal latency freshness degradation) and partial schema inconsistency. This test is to determine whether the framework can handle realistic failure cascades with multiple types of degradation, addressing the internal validity threat described in Section 9D.

6) Mild semantic drift (Day 88): Increasing the exposure metric aggregation window to 28 hours (17%) instead of the full 24 hours (100%) makes the procedure sensitive near the decision boundary, where the signal-to-noise ratio is low compared to the natural variance of the metric.

D. Configurations

Each condition is tested across four configurations:

- Config A (Static): The first baseline config for each subject, a Great Expectations-style specification of schemas, null rates, volume bounds, etc. Fixed self-written rules against which the other configs were validated.
- Config B (Profiling-assisted): Deequ-style constraint suggestion, where initially suggested constraints are generated from a snapshot of profiling, and not updated for the rest of the experiment.
- Config C (ARDQM) includes adaptive rules based on continuous profiling (Layer 1), adaptive rule synthesis (Layer 2), composite rules in Equation 4, and trust scoring with policy-aware promotion (Layer 3).
- Config D (Online drift detection): The same ADWIN-based adaptive baseline of [13] is applied separately to each profiled field. ADWIN maintains a sliding time window with variable size and issues a drift alert when the mean of the sub-window deviates at importance level α=0.01 (the default of ADWIN). When an incoming field is flagged for a drift alarm, the corresponding quality check is simultaneously flagged. This is the best single-signal adaptive result available and is strictly better than Config B with a fixed profiling snapshot. ADWIN handles univariate streams and does not natively support composite multi-signal rules, trust scoring, or policy-aware promotion.

E. Metrics and Statistical Analysis

Each configuration is evaluated on two measures for each degradation condition:

- Detection rate: fraction of degradation events captured before reaching downstream consumers.
- False-positive rate: fraction of evaluation windows not degraded but classified as degraded.

For each configuration, 30 runs with varying random seeds were performed. The results are presented as mean ± standard deviation. Statistical importance is assessed using two-sided Welch's t-tests with $\alpha = 0.05$, and effect sizes are reported with Cohen's d.

F. Statistical Methodology

Experimental unit. A simulation run is an experimental unit. A simulation run is defined as each of the 90-day runs with a new random seed. For each degradation condition, each run produces a single detection rate and a single false-positive rate. Given that there were 30 runs per condition per configuration, there were n = 30 in each cell of Table 4.

Confidence intervals. The 95% confidence intervals are calculated as $\bar{x} \pm t_{0.025,n-1} \cdot (s / \sqrt{n})$, where $\bar{x}$ is the sample mean, s is the sample standard deviation, and $t_{0.025,29} = 2.045$ is the critical value of the student's t-distribution with 29 degrees of freedom (df).

Statistical test selection. Welch's t-test is used since the configurations being compared have different variances, such as the SD of false positives in the static baseline with volume spike being 0.04 and the same SD in ARDQM being 0.01. Unlike the Student's t-test, Welch's t-test does not assume the variances of populations are equal.

It uses the Welch-Satterthwaite equation to estimate the appropriate degrees of freedom. It is two-sided, to avoid directional bias.

Effect size. Cohen's d, which can be calculated as $d = (\bar{x}C - \bar{x}A) / spooled$, where $spooled = \sqrt{((nC - 1)s^2C + (nA - 1)s^2A) / (nC + nA - 2)}$. Effects sizes larger than $d = 0.8$ are considered large effects.

The study yields twelve comparisons between each pair of configurations, considering six degradation conditions and two metrics per condition for multiple-comparison correction. Family-wise at $\alpha = 0.05$, a Bonferroni correction would require $p < 0.0042$ for importance (0.05 / 12). Because all reported p-values are below 0.001, these conclusions hold under the Bonferroni correction. Since all p-values are less than 0.001, the results of either analysis are unaffected by the Bonferroni correction. The nominal p-values for the extended scenarios (Table 7) and the sensitivity analysis, both of which were Bonferroni adjusted for this larger set of comparisons, are all less than 0.001.

Normality check. Shapiro-Wilk tests on each of the thirty-run samples in each condition-metric combination were all non-significant ($p > 0.10$), allowing us to use parametric tests. In addition, non-parametric Mann-Whitney U tests were run as a robustness check with the same result.

G. Results

Detection rates and false-positive rates at each of these conditions and configurations are found in Table 4.

TABLE 4. Detection rate and false-positive rate by degradation condition and configuration. Values are mean ± SD with 95% CI over 30 independent runs. All Config C vs. A/B/D differences are significant at $p < 0.001$ (two-sided Welch's t-test, n = 30)

Condition	Metric	Static (A)	Profiling (B)	ADWIN (D)	ARDQM (C)
Producer dropout	Detection	0.85±0.03 [0.84,0.86]	0.88±0.03 [0.87,0.89]	0.92±0.02 [0.91, 0.93]	0.97±0.02 [0.96,0.98]
Producer dropout	False-pos.	0.12±0.02 [0.11,0.13]	0.09±0.02 [0.08,0.10]	0.07±0.02 [0.06, 0.08]	0.04±0.01 [0.04,0.04]
Schema drift	Detection	0.60±0.04 [0.59,0.62]	0.74±0.03 [0.73,0.75]	0.79±0.03 [0.78, 0.80]	0.91±0.02 [0.90,0.92]
Schema drift	False-pos.	0.21±0.03 [0.20,0.22]	0.14±0.02 [0.13,0.15]	0.11±0.02 [0.10, 0.12]	0.06±0.01 [0.06,0.06]
Semantic drift	Detection	0.10±0.03 [0.09,0.11]	0.31±0.04 [0.30,0.33]	0.48±0.04 [0.47, 0.50]	0.78±0.03 [0.77,0.79]
Semantic drift	False-pos.	0.05±0.01 [0.05,0.05]	0.07±0.02 [0.06,0.08]	0.06±0.02 [0.05, 0.07]	0.03±0.01 [0.03,0.03]
Volume spike	Detection	0.95±0.02 [0.94,0.96]	0.95±0.02 [0.94,0.96]	0.95±0.02 [0.94, 0.96]	0.95±0.02 [0.94,0.96]
Volume spike	False-pos.	0.38±0.04 [0.37,0.40]	0.22±0.03 [0.21,0.23]	0.14±0.02 [0.13, 0.15]	0.05±0.01 [0.05,0.05]

H. Effect Size Analysis

Table 5 reports Cohen's d for the pairwise comparison between ARDQM (Config C) and the static baseline (Config A) on detection rate.

TABLE 5: Effect size (Cohen's d) for detection rate: ARDQM (c) vs. static (a). Values > 0.8 indicate large effects.

Condition	Cohen's d

Producer dropout	4.71
Schema drift	9.84
Semantic drift	22.67
Volume spike	0.00
Correlated failure	~9.25
Mild semantic drift	~11.07

I. Ablation Study

While ARDQM produces an improvement over the baselines as a whole (Table 4), there is no way out of the box to identify which architectural change is the source of which improvement. In Table 6, we ablate the ARDQM run through the analogous simulation harness and seed sequence on both semantic drift (where the binding constraint is detection) and volume spike (where the binding constraint is false-positive rate). Three reduced models are compared to the full model (C1):

- C2: ARDQM with no composite rules; only single-condition adaptive rules (profiling and rule evolution kept, Equation 4 disabled).
- C3: ARDQM no trust scoring: composite rules still used, but decision is presented as binary pass/fail decision, instead of being combined with the graded score (Equation (6)), thus removing the ability to use conditional serving.
- C4: ARDQM with no continuous profiling: deployments with rules created only once from the current state of the system, equivalent to the static configuration B in Table 4.

TABLE 6. Ablation of ARDQM components on semantic-drift detection and volume-spike false-positive rate. Values are mean ± SD with 95% CI over 30 runs. Simulation-based; not production-validated.

Configuration	**Semantic-drift detection**	**Volume-spike false positive.**
C1 — Full ARDQM	0.78±0.03[0.77,0.79]	0.05±0.01[0.05,0.05]
C2 — without composite rules	0.34±0.04[0.33,0.36]	0.06±0.01[0.06,0.06]
C3 — without trust scoring	0.77±0.03[0.76,0.78]	0.14±0.02[0.13,0.15]
C4 — without continuous profiling (= Config B)	0.31±0.04[0.30,0.33]	0.22±0.03[0.21,0.23]

Ablating composite rules (C2) reveals the two effects in Table 4: semantic-drift detection drops from 0.78 to 0.34, while volume-spike false positives are unaffected (0.05→0.06) since they are seen by a single behavioral feature and not composite rules. Removing trust scoring (C3) only marginally degrades semantic drift detection (0.78→0.77) and triples the false-positive rate for volume spikes (0.05→0.14), while the binary monitoring decisions would not have the option to serve conditionally (i.e., choosing the best one). Removing continuous profiling (C4) dramatically obstructs both metrics. Thus, composite rules are responsible for the majority of semantic drift gain, trust-based graded promotion is responsible for the majority of false-positive reduction, and both can be attributed to continuous profiling.

J. Discussion of Results

The most obvious gap is semantic drift detection. Static and profiling-based frameworks do not support composite rules for methodology changes that do not change the system's architecture. ARDQM's composite rule (Equation 4) detects semantic drift based on the correlation of affected metrics with correlated downstream signals and the absence of an explanatory deployment event. However, this effect size (d = 22.67) is huge due to the fact that

the semantic drift condition was designed to be invisible to structural checks (all three schema, null rate, and volume bounds hold). As such, Configs An and B have almost no detection baseline at all. Thus, comparing semantic drift to the baseline is not merely a marginal improvement in a single metric; instead, it concerns whether to employ a detection mechanism, which represents a categorical capability gap. When semantic drift arguably coincides with a structural signal in deployments (for example, a schema annotation change or a producer version bump coinciding with a methodology change), detection rate under the static baseline would noticeably increase and effect size would noticeably decrease. Thus, in this sense, the d = 22.67 result should not be interpreted as a guarantee that ARDQM outperforms alternatives by a factor of 22.67 in production but rather that composite cross-signal rules are architecturally necessary to detect methodology drift.

This was true both in and across datasets. In correlated failure data (Table 7), for example, ARDQM's detection rate of 0.89 ± 0.03 is higher than the 0.52 ± 0.04 from the static baseline and 0.69 ± 0.03 from ADWIN, which implies that our cross-signal rules enabled a higher detection rate in the failure cascade than if each stream was monitored independently. ADWIN (when applied separately to each field stream) detects more than the baseline but has the same limitation: it does not exploit the temporal correlation of the streams. For mild drift (17% aggregate shift), ARDQM did not detect the drift. These results show that small noise levels are still difficult even with composite rules, indicating that the method's sensitivity is intrinsically limited. Future work might focus on adaptive thresholds in conjunction with long observation windows.

The volume spike condition is correctly detected with a rate of 0.95 in all four frameworks, because they all have a volume-based monitor. The behavioral baseline of ARDQM instead reduces the false positive rate, from 0.38 to 0.05, with the corresponding saving in incident-response effort. The rates of false positives are reduced due to the rolling baseline computation in Equation 1, which allows ARDQM to differentiate between actual increases in data and an increase that can be classified as an anomalous spike. The 86% reduction is only in the case of the volume-spike condition. The reductions under the other three conditions are smaller, as shown by Table 4.

The remaining 22% of undetected semantic drift events occurring in Config C are due to the low magnitude of the methodology's change relative to the natural variance of the metric (σM) which results in an observed deviation score (Equation 2) being lower than the threshold of δ_t δthresh. Adaptive threshold tuning is a potential area for future work.

For producer dropout and schema drift, ARDQM's performance compared to Config B improved because ARDQM kept evolving the rules as continuous producers were added, while Config B only trained on a snapshot of the data.

However, these relative improvements, when making use of this detection method, are typically only seen under controlled conditions and may not be reproduced in production.

TABLE 7. Detection rate and false-positive rate for extended degradation scenarios (Conditions 5–6). Values are mean ± SD with 95% CI over 30 independent runs. All Config C vs. A/B/D differences are significant at $p < 0.001$ (two-sided Welch's t-test, n = 30).

Condition	Metric	Static (A)	Profiling (B)	ADWIN (D)	ARDQM (C)
Correlated failure	Detection	0.52±0.04 [0.51, 0.54]	0.61±0.04 [0.60, 0.63]	0.69±0.03 [0.68, 0.70]	0.89±0.03 [0.88, 0.90]
Correlated failure	False-pos.	0.25±0.03 [0.24, 0.26]	0.18±0.03 [0.17, 0.19]	0.13±0.02 [0.12, 0.14]	0.07±0.01 [0.07, 0.07]
Mild semantic drift	Detection	0.06±0.02 [0.05, 0.07]	0.14±0.03 [0.13, 0.15]	0.22±0.04 [0.21, 0.24]	0.41±0.04 [0.40, 0.43]
Mild semantic drift	False-pos.	0.04±0.01 [0.04, 0.04]	0.05±0.01 [0.05, 0.05]	0.05±0.01 [0.05, 0.05]	0.04±0.01 [0.04, 0.04]

K. Sensitivity Analysis

To test the robustness of the results to hyperparameters, a sensitivity analysis is performed on the size of the profiling window W, the deviation threshold δ_t δthresh, and the weights wd of the four dimensions of the trust score. Each parameter is varied while fixing the other parameters to their default values. Results are averaged over 30 runs per setting.

The *window size W* is evaluated for the values 3, 5, 7, 10, and 14 days. W = 3 should improve spike detection, even when it occurs over a very short time period. The producer dropout detection improves from 0.97 to 0.98, while the false positive rate increases from 0.05 to 0.11. The trade-off of W = 14 provides a very low false positive rate but two days of delay (0.91 to 0.84). Although W = 7 is a reasonable compromise between timely notification and low latency, the best value of W varies considerably between telemetry conditions.

Deviation threshold δ_t δthresh. Evaluate δ_t and δ_t δthresh ∈ {1.5, 2.0, 2.5, 3.0, 3.5} standard deviations. This means that lower thresholds generally increase semantic drift detection (0.78 to 0.87 for δ_t δthresh = 1.5) but also all other false positive scenarios. Only when the threshold is increased to δ_t δthresh = 3.5 we see a reduced semantic drift detection rate (0.62). As this relationship is monotonic and clear, we can use ROC analysis to find a threshold appropriate for an application.

Three sets of dimension weights wd are tested: (1) uniform weights wd = 0.25 spanning all dimensions. (2) behavioral-heavy weights (wbehavioral = 0.40, all other weights 0.20) or (3) semantic-heavy weights (wsemantic = 0.40, all other weights 0.20). The semantic-heavy profile is considerably better at separating semantic drift (the mean τ gap between drifted and non-drifted windows improves from 0.12 to 0.19) and slightly worse at separating producer dropout. This indicates that the weight configuration requires adaptation for different downstream tasks, which was already anticipated by the model's policy-aware framework.

Overall, our results suggest the qualitative benefits of ARDQM over these baselines hold for a reasonable range of conditions, and the size of the improvements varies based on which environment the ARDQM is being used in.

L. Worked Example: Composite Rule Detection of Semantic Drift

The next example of semantic drift in the simulation helps to further explain how this composite rule mechanism detects semantic drift in practice.

On Day 60, the engagement metric session_count changes from a rolling 24-hour window to a rolling 48-hour window. The schema remains the same. The field's type name and nullability have not changed. Static rule set A, checking schema, null rate and volume constraints, passes its checks. There is no detectable drift.

Under ARDQM, this yields the composite rule R^c:

- c_1: The mean of session_count deviates from its 7-day rolling (moving) average by more than δ_t, where δthresh = 2.5σ.
- c_2: The ratio of sessioncount to a correlated downstream metric (dailyactive_users) differs from its historical ratio baseline by more than 2.0σ.
- c_3: No schema version change or producer deployment event was recorded in the metadata log in the last 48 hours.

An example of such a shift on Day 60 would be the doubling of the sessioncount (the 48-hour windows account for roughly twice the number of events), with the mean shift being approximately 4.1σ, thereby satisfying condition c_1. Condition c_2 would also be satisfied, as sessioncount/dailyactiveusers would shift by approximately 3.8σ, while dailyactiveusers does not vary. Condition c_3 is satisfied if no deployment or schema event accounts for the change.

As the composite rule is induced ($\rho = 3/3 = 1.0$), the multi-action response function ϕ considers the log an extreme semantic anomaly, quarantining it for other data consumers with $\theta High \geq 0.90$. For consumers with a lower threshold, the data annotated with a flag will be used.

This illustrates why no single-condition rule detects semantic drift. Volume, schema, and null-rate do not detect drift by themselves. Only the composite rule, which correlates the metric drift with a related signal and states that no deployment event justifies the drift, detects the drift.

9. LIMITATIONS AND THREATS TO VALIDITY

Every quantitative result in Section VIII, including the ablation and sensitivity studies, was conducted in simulation and has not been deployed or validated on a production telemetry system. The text below should be read with this constraint in mind throughout, not just in the external-validity subsection: the model has never been deployed or validated on any production telemetry system. These limits fall into four distinct categories.

A. Cold-Start Behavior

This profiling layer (Equation 1) requires a minimum of W time windows to learn a historical profile of the data. During this cold-start period, there is not enough time series history to distinguish normal behavior from an anomaly. Post-initialization, ARDQM degrades to static rules, offering no advantage over Config A. The cold start phase for new telemetry producers can last days to weeks, depending on the size of the window parameter W used during collector initialization. In future work, we will explore strategies to reduce cold start in ARDQM, such as knowledge transfer from similar telemetry producers or bootstrapping from schema-derived priors.

B. Computational Overhead

Ongoing profiling, maintaining the rolling baseline, and evaluating the composite rules are all linear in the number of producers, number of fields, and complexity of the rule. There is $O(k \cdot N)$ profiling overhead per window, where k is the number of fields being profiled and N is the number of records in the aggregated window. The per-rule evaluation of composite rules has O(n) overhead. At scale, this requires dedicated infrastructure, and quantifying underproduction workloads is left for future work.

C. Rule Drift Risk

On the other hand, adaptive rule evolution (Equation 3) risks the rules overfitting to prevent continual deterioration. The drift threshold δ_r, the minimum evidence requirement n_min, the safety constraints in Algorithm 1, and the rollback mechanism address this concern. However, slow degradation over many windows still allows the source to be absorbed into the rolling baseline before being flagged. Periodic manual review of rule evolution history and hard lower bounds on important quality parameters are suggested as operational checks.

D. Threats to Validity

Construct validity. The detection rate and the false-positive rate as a proxy measure for decision quality in real life. The impact of missing a detection in production would depend on the downstream decision. This simulation does not currently model these impacts, but it is important to consider them for the future.

Internal validity. The simulation uses synthetic degradation conditions with controlled injection schedules. In contrast to real-world failures, which often have correlated effects (e.g., a network partition is most likely to immediately result in producer dropout, late arrivals, and schema drift), the first four types of failure conditions are independently injected at non-overlapping time points. The correlated-failure and mild-semantic-drift conditions are meant to reflect real-world failures, which relax the independence assumption to some extent. The Poisson arrival model also assumes independence, which may underestimate correlated failure cascades. Reporting the results of 30 independent runs with statistical importance can reduce this concern.

External validity. The testing is done entirely in simulation and not on a production telemetry infrastructure. Though the simulation parameters for scenarios are based on patterns in the literature on distributed systems [1] and experimentation [15, 16], the lack of experimental validation means that these results should not be taken as generalizable to production systems. Instead, they should be taken as an estimate of what the framework could achieve in an idealized environment. Empirical evaluation is the next big challenge using production telemetry traces.

Fair comparison. Table 1 compares architectures, not production performance. Existing systems have been in production usage for over a decade, have been subjected to many workloads, have large user communities, and are compatible with existing systems. In contrast, ARDQM is a proposed simulation system with no production usage history, as Table 1's last few rows show. The one caveat to this analysis is that while this table shows gaps in existing tools, ARDQM is not necessarily better.

Reproducibility. The authors have not made the code available to reproduce the simulation study presented in the paper. All simulations, data decay schedules, random seed sequences, and baseline implementations were conducted with standard, open-source Python libraries: SimPy 4.0, NumPy, and SciPy. Bibliographic information on these libraries is provided in Table 3, and implementation details are provided in section 5-7. The algorithm itself is given in (Algorithm 1). Since the algorithm as described was not guaranteed to run, and since the results depend on the exact specification used, it is possible to implement the algorithm in a way that behaves differently from the paper's description.

10. CROSS-INDUSTRY RELEVANCE

The model is a step towards a dynamic, context-sensitive quality system. Modern telemetry systems tend to be distributed, lifecycle-evolving, and much more tightly coupled to operational aspects [2]. The model allows the definition of adaptive quality requirements, maintaining governance and auditability, which classical static models do not allow.

These principles have been generalized in other areas:

In streaming or media systems, where user engagement is measured through telemetry, silent drift can bias recommendations and A/B testing [16].

- On cloud platforms, telemetry can be used for reliability analysis, performance tuning and fault diagnosis, but is incomplete if producers do not report on delays and faults they observe.
- Quality management in IoT and other systems of connected devices requires adaptive quality logic, as the behavior at the sources is not deterministic and the heterogeneity of connected devices compounds the challenge of maintaining consistent data quality at scale [17].
- Machine-generated data is especially important in the healthcare and life sciences fields because it directly affects patient safety, and existing work on the dimensions and methods of health record data quality assessment confirms that the same structural, behavioral, and semantic validation concerns that ARDQM addresses arise in clinical data environments [18].
- The quality of financial systems is critical for financial reporting purposes, particularly in terms of the thresholds of relevance.

In all cases, though, lower-quality telemetry likely leads to lower-quality choices; it also requires domain-specific adaptations and empirical evaluations.

11. CONCLUSION AND FUTURE WORK

If QoS guarantees are required for distributed telemetry systems, their scale, heterogeneity of nodes, and sensitivity to operational conditions require adaptive solutions that evolve in parallel with the data's behavior.

The three key contributions of ARDQM are as follows. First, the conceptual model of continuous profiling, adaptive rule composition, and graded promotion is presented in a three-layer architecture. Second, the mathematical

model of trust scoring with policy-controlled weights and law-controlled thresholds. Third, the composite rule includes conditions that look for multiple signals and assess severity across multiple tiers. Statistically meaningful gains have been demonstrated in controlled discrete-event simulation over non-adaptive static rules, static profiling-improved rules, and rules with online drift detection (ADWIN). The effect size is largest for semantic drift (d = 22.67, again, a categorical capability gap engineered by the simulation design rather than by a production deployment) and for false positive reduction when there are sudden increases in traffic volume (86%, only the volume-spike condition). These results are specific to the simulated profile and degradation behaviors, and our next step is to validate these findings in production. These results are specific to this simulation, and their impact on deployed systems remains to be seen.

Limitations include cold-start behavior, overhead, drifting learned rules, and a lack of benchmarking on production data. Future work could be validated on production telemetry traces, perform adaptive threshold tuning using ROC analysis, model cascading failures that are correlated beyond the extended one considered in this paper, or benchmark the computational cost of the algorithm on production data.

Overall, the model provides a reference model for telemetry quality management, conceptualizing it as a lifecycle-spanning and adaptive rather than a static process that takes place in the system under test.

DECLARATION OF THE USE OF AI TOOLS

This paper did not use AI tools.

REFERENCES

1. M. Kleppmann, "Designing Data-Intensive Applications," O'Reilly Media, Sebastopol, CA, 2017.
2. B. Burns, J. Beda, K. Hightower, "Kubernetes: Up and Running," 2nd Ed., O'Reilly Media, Sebastopol, CA, 2018.
3. B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, C. Shanbhag, "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure," Google, Inc., Tech. Rep., 2010.
4. T. C. Redman, "The Impact of Poor Data Quality on the Typical Enterprise," Communications of the ACM, Vol. 41, No. 2, 1998, pp. 79–82, doi: 10.1145/269012.269025.
5. C. Batini, C. Cappiello, C. Francalanci, A. Maurino, "Methodologies for Data Quality Assessment and Improvement," ACM Computing Surveys, Vol. 41, No. 3, 2009, pp. 1–52, doi: 10.1145/1541880.1541883.
6. L. L. Pipino, Y. W. Lee, R. Y. Wang, "Data Quality Assessment," Communications of the ACM, Vol. 45, No. 4, 2002, pp. 211–218.
7. R. Y. Wang, D. M. Strong, "Beyond Accuracy: What Data Quality Means to Data Consumers," Journal of Management Information Systems, Vol. 12, No. 4, 1996, pp. 5–33, doi: 10.1080/07421222.1996.11518099.
8. A. Campbell et al., "Great Expectations: A Shared, Open Standard for Data Quality," 2021, https://greatexpectations.io (accessed: 2024).
9. S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biessmann, A. Grafberger, "Automating Large-Scale Data Quality Verification," Proceedings of the VLDB Endowment, Vol. 11, No. 12, 2018, pp. 1781–1794.
10. B. Moses, L. Gavish, M. Vorwerck, "Data Quality Fundamentals," O'Reilly Media, Sebastopol, CA, 2022.
11. E. Breck, N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data Validation for Machine Learning," Proceedings of the 2nd Conference on Machine Learning and Systems (MLSys), Stanford, CA, USA, 2019.
12. E. S. Page, "Continuous Inspection Schemes," Biometrika, Vol. 41, No. 1/2, 1954, pp. 100–115, doi: 10.1093/biomet/41.1-2.100.
13. J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, "A Survey on Concept Drift Adaptation," ACM Computing Surveys, Vol. 46, No. 4, 2014, pp. 1–37, doi: 10.1145/2523813.
14. S. Rabanser, S. Günnemann, Z. Lipton, "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift," Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 2019.
15. B. Beyer, C. Jones, J. Petoff, N. R. Murphy, "Site Reliability Engineering," O'Reilly Media, Sebastopol, CA, 2016.
16. R. Kohavi, D. Tang, Y. Xu, "Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing," Cambridge University Press, Cambridge, UK, 2020.
17. L. Atzori, A. Iera, G. Morabito, "The Internet of Things: A Survey", Computer Networks, Vol. 54, No. 15, 2010, pp. 2787–2805.
18. N. G. Weiskopf, C. Weng, "Methods and Dimensions of Electronic Health Record Data Quality Assessment," Journal of the American Medical Informatics Association, Vol. 20, No. 1, 2013, pp. 144–151.
19. D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. Crespo, D. Dennison, "Hidden Technical Debt in Machine Learning Systems," Proceedings of the 28th Conference on Neural Information Processing Systems (NeurIPS), Montreal, Canada, 2015.
20. N. Polyzotis, S. Roy, S. E. Whang, M. Zinkevich, "Data Lifecycle Challenges in Production Machine Learning: A Survey," ACM SIGMOD Record, Vol. 47, No. 2, 2019, pp. 17–28.

21. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, "Experimentation in Software Engineering," Springer, Berlin, Germany, 2012.
22. Z. Dehghani, "Data Mesh: Delivering Data-Driven Value at Scale," O'Reilly Media, Sebastopol, CA, 2022.
23. E. Rahm, H. H. Do, "Data Cleaning: Problems and Current Approaches," IEEE Data Engineering Bulletin, Vol. 23, No. 4, 2000, pp. 3–13.
24. Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, N. Tang, "Detecting Data Errors: Where Are We and What Needs to Be Done?" Proceedings of the VLDB Endowment, Vol. 9, No. 12, 2016, pp. 993–1000.
25. A. Jones, "Driving Data Quality with Data Contracts," O'Reilly Media, 2023.
26. C. Sridharan, "Distributed Systems Observability," O'Reilly Media, Sebastopol, CA, 2018.