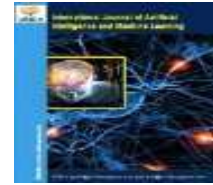




International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: <https://www.svedbergopen.com/>



Research Paper

Open Access

A Sensor Data Quality and Validation Framework For Large-Scale Industrial Monitoring: Detecting Faulty, Drifting, And Missing Telemetry Before It Reaches Analytics

Jasvitha Buggana

Independent Researcher, India. E-mail: jasvitha.work@gmail.com

Abstract

Bad sensor data rarely fails loudly. A drifting transmitter or a silently dropped signal keeps producing plausible values, so dashboards, alarms, and models built on that data degrade without triggering any obvious fault. Analytics teams continue to trust numbers that no longer reflect physical reality, and the delay is discovered only when an operational event forces a retrospective look at the trend. This paper proposes a sensor data quality and validation framework that screens industrial telemetry for faults, drift, and missingness before it reaches downstream analytics, rather than cleaning it afterwards. The framework is specified as an ordered pipeline: structural validation at ingestion, physical plausibility bounds derived from instrument class, drift detection against each tag's own rolling baseline rather than a fixed limit, staleness and value-repetition detection against expected reporting frequency, and a quality status assigned per reading that gates what is allowed to inform a dashboard, model, or alert. Two discriminations carry the design. Sensor fault is separated from genuine process change by testing whether an anomaly is isolated to one tag or corroborated across physically related tags. Gradual drift is separated from abrupt fault by persistence and pattern shape rather than instantaneous magnitude. The framework is a proposed design. It has not been implemented or evaluated against live or historical data, and no detection rate, false-alarm rate, or detection latency is reported. Section 7 specifies the evaluation that would be required to establish them.

Keywords: sensor data quality, industrial telemetry validation, sensor drift detection, missing data detection, fault detection, condition monitoring, SCADA data quality, historian data, anomaly detection, data validation framework, predictive maintenance, industrial IoT, time-series validation

I. Introduction

Industrial monitoring rests on an assumption that is rarely tested: that a value arriving from a sensor is a measurement. Historians are built to record faithfully, and they record faithfully whatever the instrument reports, including values the instrument reports after it has ceased to measure anything meaningful. A transmitter that has drifted out of calibration continues to emit numbers. A communication path that has stopped updating leaves a last known value in place, and a value in place is read downstream as a current value. The failure is quiet, and quietness is the whole problem. Equipment faults announce themselves through alarms because alarms are configured to detect them. Data faults have no equivalent, because the systems that would raise the alarm are the systems consuming the compromised data. The dashboard renders green. The condition-monitoring model receives its input and produces an output. The alarm does not fire, and the absence of an alarm is interpreted, correctly under the system's own logic and incorrectly in fact, as evidence that nothing is wrong.

This produces a specific and under-addressed class of operational risk. Where a loud failure causes a false alarm and wastes attention, a quiet failure causes a missed event and consumes the interval in which the event could have been prevented. The cost is not the bad reading. The cost is the decision that was not taken during the weeks in which the data appeared normal.

Current practice addresses this mostly after the fact. Static threshold alarms in the control layer catch values outside operational limits, which is a different question from whether the instrument is healthy. Periodic manual calibration checks catch drift at the calibration interval, which may be months. Data cleaning by

analytics or data engineering teams catches problems once someone downstream has noticed that a model output or a dashboard looks wrong. Each of these is useful and none of them is early.

The limitation shared by all three is that they are triggered by a human noticing, or by a calendar, rather than by systematic screening. That makes them structurally worst at exactly the failure modes that matter most here. Slow drift is invisible to a threshold because it stays inside the threshold by construction. A frozen signal is invisible to a range check because the frozen value is usually a plausible one. Both are the failure modes least likely to prompt a human to look.

The proposal in this paper is to treat data quality validation as a layer, positioned between raw telemetry and analytics, rather than as an upstream alarm or a downstream cleanup. The question the layer asks is different from the question a threshold asks. A threshold asks whether this value crossed a fixed limit. The validation layer asks whether this value, and this stream's recent behaviour, remain statistically consistent with a healthy instrument, and it asks it before the value is permitted to inform anything.

The contribution is the composition rather than any individual detector. CUSUM, EWMA, range checks, and staleness detection are all established. What is proposed here is their arrangement into a pre-analytics gate that treats faults, drift, and missingness as one problem with three signatures, assigns an explicit quality status rather than a binary verdict, and carries that status forward so downstream consumers can distinguish trusted data from compromised data instead of treating all of it as equally reliable.

Section 2 positions the work and states its boundary against the author's prior paper. Section 3 characterizes the three failure modes and where they arise. Section 4 specifies the pipeline. Sections 5 and 6 address the two hard discriminations, sensor fault versus process change and drift versus noise, together with alarm-flood control. Section 7 gives the evaluation design that this paper does not itself carry out. Sections 8 and 9 state the limits and conclude.

2. Background and positioning

2.1 Data quality as a defined property

Treating data quality as a property with dimensions rather than as a vague adequacy judgement follows Wang and Strong, whose framework established that quality is defined relative to the consumer's use rather than intrinsically [1]. That orientation matters here: a value adequate for a trend display may be inadequate for a model input, which is why the framework attaches a status to data and lets the consumer decide, rather than making a single global accept or reject. The same consumer-relative view underpins the data quality vocabulary standardised in [12, 13].

Teh, Kempa-Liehr, and Wang provide a systematic review of sensor data quality specifically, covering error taxonomies and detection approaches across sensing domains [2]. Their synthesis confirms the failure modes treated here as the recurring ones, and also confirms the pattern this paper responds to, namely that detection techniques are typically studied in isolation rather than composed into an operational gate.

2.2 Fault detection, change detection, and drift

The detection machinery is well established and is used here rather than invented. Isermann surveys model-based fault detection and diagnosis, including the distinction between a fault in the process and a fault in the instrument observing it, which is the discrimination Section 5 operationalizes [3]. Chandola, Banerjee, and Kumar provide the general anomaly detection taxonomy, including the point, contextual, and collective categories that map onto the spike, drift, and staleness signatures treated here [4].

For sequential change detection, the framework draws on two classical methods. Page's cumulative sum procedure detects a sustained shift in a process mean, accumulating evidence across observations rather than judging each in isolation [5]. Roberts' geometric moving average, now generally the exponentially weighted moving average, provides a smoothed baseline that weights recent observations more heavily [6]. Both are chosen for interpretability: an operator asked to act on a flag needs to be able to see why it was raised, which favours these over an opaque learned detector.

Gama et al. survey concept drift adaptation in data streams [7]. Their framing is adjacent but distinct in an important way. That literature is largely concerned with a model whose input distribution has genuinely changed and which must adapt. The concern here is the opposite case: the input distribution has changed because the instrument is failing, and adaptation is exactly the wrong response, because a framework that adapts to drift will silently normalize it.

2.3 Contribution boundary against the author's prior work

The author has previously published on anomaly detection in industrial pumps using streaming sensor data, addressing the detection of developing equipment faults such as cavitation, bearing degradation, seal wear, and rotor imbalance from rotating-equipment telemetry [8].

The boundary between that work and this one is worth stating precisely, because both concern anomalies in industrial sensor streams and a reader could reasonably expect overlap.

That paper is about the asset. Its subject is the pump, its anomalies are equipment faults, and the sensor stream is the instrument through which those faults are observed. It assumes, as any such work must, that the telemetry is a faithful representation of the machine.

This paper is about the instrument. Its subject is the sensor and the path from sensor to historian, its anomalies are measurement faults, and the equipment is the context that determines what a plausible measurement would be.

The two are complementary and sequential rather than competing. The validation layer proposed here is logically upstream of the detection described in [8]: an equipment anomaly detector trained on drifting telemetry will learn the drift as a legitimate operating regime, and one fed a frozen signal will read stability where there is none. The prior work is not restated here, and no equipment fault detection method is proposed in this paper.

2.4 The gap

Detection techniques for range violation, sustained mean shift, and missingness are individually mature. What the literature treats less systematically is their composition into a gate that sits before analytics, handles the three failure modes as a single operational concern, distinguishes an instrument fault from a real process change as a first-class requirement rather than an afterthought, and emits a graded quality status that downstream consumers can act on differentially. That composition is what this paper proposes.

3. Failure modes and where they arise

3.1 The drift failure mode, illustrated

The following scenario is an illustrative construction, not an incident report. It is included because the drift failure mode is difficult to convey in the abstract, and its economics are the reason the framework is worth building.

Consider a vibration sensor on a critical pump at a compressor station, where the sensor's mounting bracket has loosened. Over roughly six weeks the baseline reading creeps upward from about 2.5 mm/s to about 4.8 mm/s. The alarm threshold is set at 7.0 mm/s. No alarm fires at any point, because no reading ever approaches the threshold. The historian records every value as normal. The condition-monitoring dashboard shows green status throughout. The maintenance team, reading a healthy system, defers a scheduled bearing inspection that would have caught early wear.

In the seventh week the bearing fails, forcing an unplanned shutdown. What would have been a four-hour planned maintenance task in a scheduled window instead becomes an unplanned outage of more than two days, carrying lost production, emergency parts sourcing, and overtime labour. A post-incident review finds that the drift was visible in the raw trend data throughout. It simply never crossed a static threshold, so nothing directed anyone to look at it.

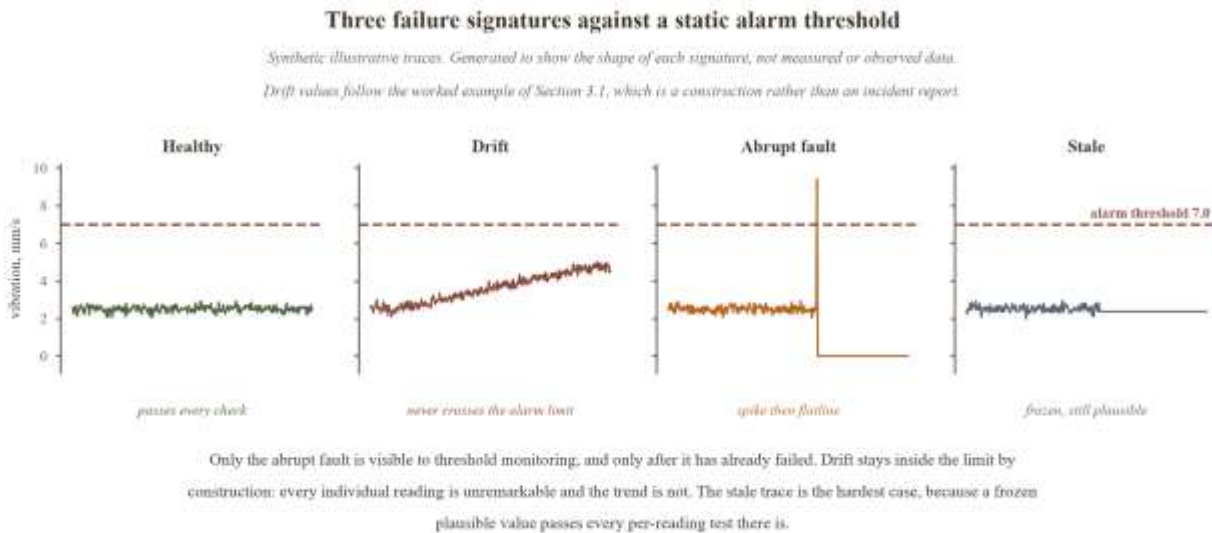


Figure 1. The three failure signatures against a static alarm threshold. Traces are synthetic and illustrative, generated to show the shape of each signature rather than to report observed data. The drift case is the one that defeats threshold monitoring by construction: every individual reading is unremarkable, the trend is not, and no fixed limit is ever crossed.

The point the scenario makes is structural rather than anecdotal. Threshold monitoring asks a question about each reading independently. Drift is a property of a sequence. No amount of care in setting the threshold makes a per-reading test detect a trend that stays inside it.

3.2 The three modes

Faulty readings arise at the instrument and its communication path: failing transmitters, electrical noise, and communication errors between field device and historian. They present as spikes, flatlines, or physically implausible values such as negative pressure or temperature excursions outside equipment limits.

Drift arises gradually, most often on mechanically mounted or calibration-sensitive instruments such as vibration sensors, flow meters, and temperature transmitters. A slow calibration loss or physical shift, a loosened mount, a fouled element, an aging component, causes readings to diverge progressively from ground truth without crossing any static alarm limit.

Missing or late data arises at network and communication boundaries: gateway or historian connectivity drops, controller communication timeouts, and scheduled system maintenance. The dangerous form is not an obvious gap but a stream that has stopped updating while its last value remains readable, so that a stale value is consumed as a current one.

Problem type	Signature	Detecting check	Why the usual control misses it
Faulty: spike, flatline, out of range	Abrupt, often physically impossible	Range and plausibility bounds per instrument class	PLC alarms are set to operational limits, not sensor-health limits
Faulty: comms garbage, wrong type, bad timestamp	Structurally malformed record	Schema and structural validation at ingestion	Historian records what it receives without judging validity
Drift: gradual, inside alarm limits	Sustained baseline movement, each reading unremarkable	Rolling baseline comparison, mean, variance, trend slope	Threshold test is per reading; drift is a property of the sequence
Drift versus real process change	One tag diverging from physically related tags	Cross-correlation against related tags	No single-tag test can distinguish them

Missing: feed stops updating	No new values against expected cadence	Expected reporting frequency per tag	Absence of data is not itself a value, so nothing is evaluated
Missing: frozen value repeats	Identical value beyond expected variance window	Value repetition check	A repeated plausible value passes every per-reading test

Table 1. Failure modes, their signatures, and the check assigned to each. The final column is the argument for the framework: each mode defeats the control that would ordinarily be expected to catch it.

3.3 Signal sources in scope

The framework targets signal types common to large-scale industrial monitoring, particularly in oil and gas and process industries

Source category	Signals	Origin	Dominant problem
Process variables	Temperature, pressure, flow rate, level	Transmitters via PLC and DCS, logged to historian	Drift on calibration-sensitive instruments; implausible values on transmitter failure
Rotating equipment	Vibration, RPM, motor current, bearing temperature	Pumps, compressors, turbines	Drift from mounting and mechanical change; feeds condition monitoring directly
SCADA status and control	Valve position, pump run status, setpoints	Control layer	Staleness read as steady state
Environmental and safety	Ambient temperature, gas detection	Safety instrumentation	Staleness and undetected fault carry safety consequence

Table 2. Signal sources in scope and the failure mode each is most exposed to. The rotating-equipment row is the one that feeds predictive maintenance, which is where undetected drift compounds into a wrong maintenance decision.

4. The validation framework

The framework is a pipeline of checks positioned between telemetry ingestion and analytics, not a single model. In the layered enterprise-control reference model [10], it occupies the boundary between the data collection layer and the analytics and operations-management layers above it. Ordering matters: each stage assumes the preceding stage has passed, so structural validity is established before physical plausibility is tested, and plausibility before statistical behaviour.

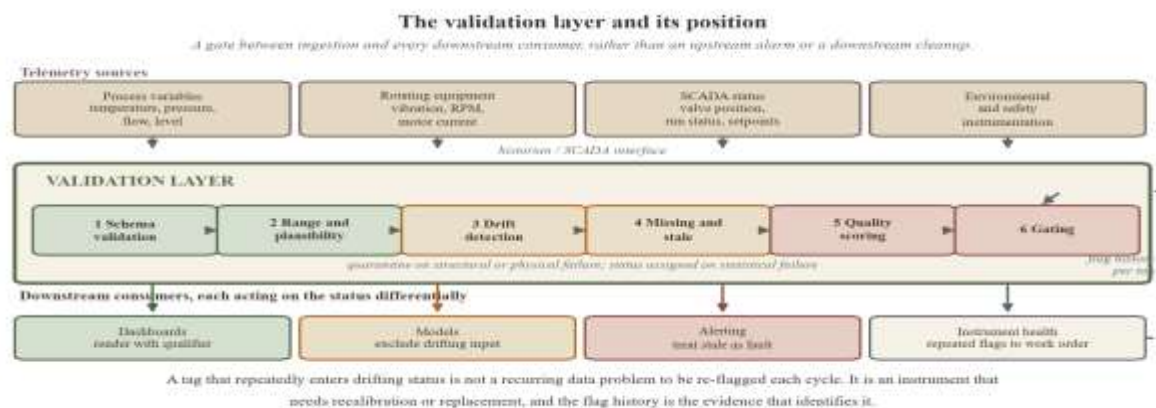


Figure 2. The validation layer and its position. The gate sits between ingestion and every downstream consumer, so no dashboard, model, or alert receives a reading that has not been assigned a quality

status. The dashed return path is the feedback of Section 4.6, which converts repeated flags on one tag into an instrument-health signal rather than a repeated per-reading nuisance.

4.1 Ingestion and schema validation

Each arriving reading is checked for structural integrity: correct data type, expected units, a valid timestamp, and a recognized tag or sensor identifier. Where the transport already carries a quality attribute, as in the industrial interoperability standard [11], that attribute is consumed as an input to this stage rather than treated as sufficient on its own, since it reports transport and device state rather than measurement validity. Malformed records are flagged immediately and routed to a quarantine log rather than passed forward.

This stage exists to prevent a category error later. A record with a corrupt timestamp can pass every statistical test and still corrupt a baseline, because the baseline is a function of ordering.

4.2 Range and plausibility checks

Each signal is checked against physically plausible bounds specific to its instrument class. Pressure cannot be negative. Vibration cannot exceed a physically realistic ceiling for the equipment class.

The distinction from control-layer alarms is the point of this stage. A PLC range check is configured to operational limits, the band within which the process should run. A plausibility check is configured to instrument-health limits, the band within which a functioning sensor can physically report. A value can be well inside the first and impossible under the second, and it is precisely the sensor-failure spikes and communication garbage that occupy that region.

4.3 Drift detection

For each stream the framework maintains a rolling statistical baseline comprising mean, variance, and rate of change over a trailing window, and compares incoming values against that baseline rather than against a fixed threshold. A signal showing a statistically significant sustained deviation from its own recent behaviour, while remaining inside operational alarm limits, is flagged as a drift candidate.

CUSUM [5] is appropriate where the concern is a sustained shift in mean, since it accumulates evidence rather than testing each observation independently. An EWMA baseline [6] provides the smoothed reference. A lightweight regression on trend slope over the window gives a directly interpretable drift rate. All three are chosen over higher-capacity learned detectors for a specific operational reason: an operator asked to raise a work order needs to see the basis for the flag, and interpretability is therefore a functional requirement rather than a preference.

4.4 Missing and stale data detection

The framework tracks expected reporting frequency per tag. A stream that stops updating, or that repeats an identical value beyond an expected variance window, is flagged as stale rather than allowed to be read as valid current data.

The value-repetition check is what addresses the more dangerous form of this mode. An absent stream is at least absent. A frozen stream presents a plausible value at the expected cadence and is indistinguishable from a genuinely stable process by any test that does not model expected variance.

4.5 Quality scoring and gating

Each reading or stream is assigned a status, and the status rather than a binary verdict is what propagates.

Status	Meaning	Gating action
Valid	Passed all checks	Flows through to analytics, dashboards, and alerting unaltered
Suspect	Anomalous but not attributable to a specific failure mode	Passed with a visible quality tag, or withheld from model input, per consumer policy
Drifting	Sustained baseline deviation within operational limits	Passed with quality tag; accumulates to a review queue rather than an operator alarm
Stale	Not updating, or frozen beyond expected variance	Withheld from consumers that would read it as current

Invalid	Structurally malformed or physically impossible	Quarantined, never passed forward
---------	---	-----------------------------------

The taxonomy deliberately parallels the device-diagnostic status categories long established for field instrumentation [9], which separate failure, function check, out-of-specification, and maintenance-required rather than reducing device health to a single bit. The framework extends that logic from the device to the data path.

Table 3. Quality status taxonomy and gating policy. The graded status is deliberate: a binary pass or fail forces a choice between discarding usable data and admitting compromised data, and the middle statuses are where most real conditions fall.

The design intent is that downstream consumers can act differentially. A trend display may reasonably render drifting data with a visual qualifier. A model retraining pipeline should probably exclude it. A safety-related alarm path should treat stale data as a fault condition rather than as a steady reading. None of those distinctions survives a single global accept or reject.

Where flagged data must be passed for continuity, it may be interpolated, but only with an explicit uncertainty flag attached, so that an interpolated value is never indistinguishable from a measured one.

4.6 Feedback and logging

Every flag is logged with the identity of the check that raised it. This produces an auditable record of data quality over time, which serves two purposes: root-cause investigation after an event, and longer-term instrument health tracking.

The second is the more valuable and the more often omitted. A tag that repeatedly enters drifting status is not a recurring data problem to be re-flagged each cycle. It is an instrument that requires recalibration or replacement, and the framework's log is the evidence that identifies it.

5. Discriminating sensor fault from process change

This is the discrimination on which operational credibility depends. A framework that flags real process changes as instrument faults will be switched off.

5.1 Corroboration across physically related tags

The distinguishing signal is not in the tag itself. It is in the tag's relationship to its physical neighbours.

A genuine process change propagates. A real pressure increase in a vessel should coincide with plausible movement in correlated flow, level, or downstream pressure. A real temperature rise should be consistent with related thermal and flow signals. The physics couples them, so a real event appears as a multi-tag, physically consistent shift.

An instrument fault does not propagate. A drifting or failing sensor diverges from its physically correlated neighbours while the rest of the unit behaves normally. The anomaly is isolated.

The framework therefore treats an isolated, uncorrelated anomaly as a likely instrument problem, and a multi-tag physically consistent shift as a likely real process event, using known process relationships such as mass balance and upstream-downstream coupling as a soft constraint rather than a hard rule. The softness is necessary: not every tag has a conveniently correlated neighbour, and a rule that requires one would leave isolated instruments unprotected.

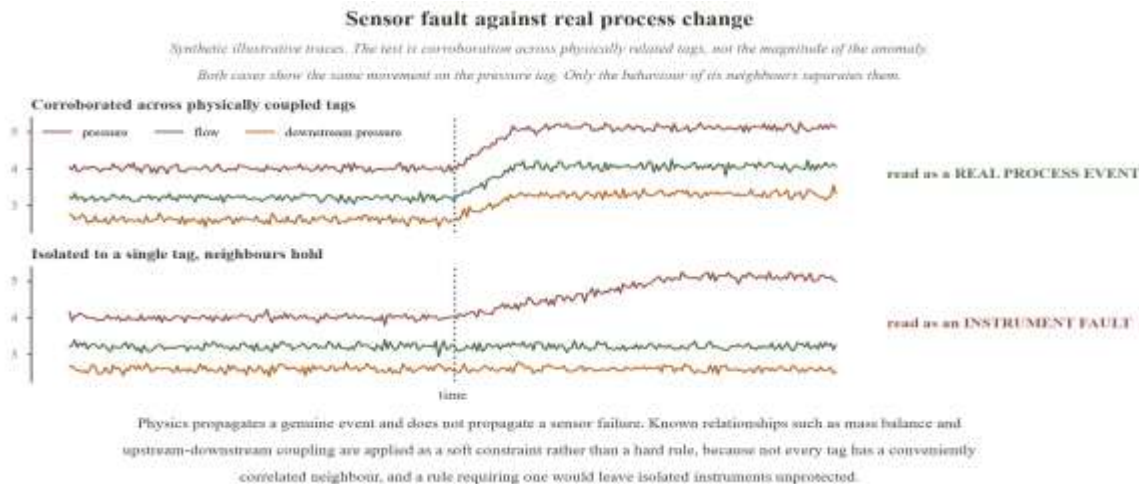


Figure 3. Sensor fault against real process change. Traces are synthetic and illustrative. In the upper case the anomaly is corroborated across physically coupled tags and is treated as a process event. In the lower case a single tag diverges while its neighbours hold, and is treated as an instrument problem. The test is corroboration, not magnitude, and the two cases can be identical in magnitude.

5.2 Drift against abrupt fault

These are separated by rate of change and pattern shape rather than by magnitude.

An abrupt fault, a wire break, a transmitter power loss, an instrument failure, presents as a sharp discontinuity, a flatline, or a spike outside physically plausible range. It is detectable almost immediately by rate-of-change and bounds checks.

Drift presents no informative instantaneous signal. Detection requires tracking the baseline, rolling mean and trend slope, over a window measured in days to weeks, and flagging a sustained statistically significant deviation while each individual reading remains unremarkable.

The distinguishing property is persistence. A fault is an abrupt and often unphysical jump. Drift is a gradual and physically plausible creep that becomes suspicious only when accumulated across time.

6. Thresholds and alarm-flood control

6.1 Per-tag threshold derivation

Thresholds are derived per tag from that tag's own historical behaviour, using percentile-based or standard-deviation-based bounds calculated over a rolling baseline period, rather than a single fixed value applied across all sensors of a type.

The reason is empirical: two nominally identical instruments on different equipment can exhibit materially different normal variance, because their mounting, process conditions, and duty differ. A global threshold calibrated to the noisier of the two is blind on the quieter one, and calibrated to the quieter one it floods on the noisier.

Where available, thresholds are additionally informed by manufacturer specifications and known process operating ranges, with one ordering rule: a statistical bound never overrides a hard physical impossibility. A value outside physical possibility is invalid regardless of what the tag's own history suggests is normal.

6.2 Preventing alarm flooding

A validation layer that produces more alerts than it prevents incidents is a net loss. Four mechanisms operate together.

Minimum persistence. A deviation must be sustained over a minimum duration before it is flagged, so single-sample noise raises nothing.

Graded status rather than binary alarm. Most conditions become a visible suspect or drifting tag rather than an operator-facing alarm, per Table 3.

Root-cause grouping. Correlated flags arising from one underlying cause, for example an upstream sensor failure that would otherwise cascade into flags on every downstream tag, are grouped into a single event rather than emitted separately.

Severity routing. An active sensor failure feeding a safety-relevant alarm path surfaces immediately. Low-severity drift and staleness accumulate into a review queue.

The intent is that urgency is preserved where it is warranted and suppressed where it is not, and that the default destination for a data quality finding is a queue rather than a control-room alarm.

7. Evaluation design

No evaluation has been performed. This section specifies what would be required, in enough detail to be executed, because a proposed framework without a stated evaluation design is not assessable.

7.1 Establishing ground truth

The central difficulty is that operational sensor data is unlabeled. Two routes give labels.

Synthetic fault injection. Take clean historical data from periods with no known instrument problems, inject drift, spikes, flatlines, and dropouts with known onset times and magnitudes, and measure whether and when the framework flags them. This gives complete ground truth and full control over the difficulty of the test, at the cost of realism in the fault model.

Historical replay against known incidents. Identify periods containing documented instrument problems, calibration records, maintenance work orders, or post-incident reviews, replay them, and compare framework flags against the documented record. This gives realism at the cost of sparse and imprecisely timed labels.

Both are needed. Synthetic injection establishes detection characteristics; historical replay establishes whether those characteristics survive contact with real data.

Metric	Definition	Requires	Why it matters
Detection rate by failure type	Share of injected or documented faults flagged, separated into faulty, drifting, and stale	Labeled set	A single aggregate rate conceals that drift is the hard case
False-alarm rate	Flags raised on periods with no known fault	Labeled clean periods	Determines whether the layer is operationally tolerable
Detection latency	Interval from fault onset to flag	Known onset times	The entire value proposition is earlier detection, so this is the primary metric
Latency advantage	Framework detection time against the time the issue was actually caught operationally, or never	Historical incidents	Measures benefit over current practice rather than against nothing
Discrimination accuracy	Correct separation of instrument fault from real process change	Labeled process events	Tests the Section 5 claim directly
Flag volume per tag per period	Operator-facing load after grouping and routing	Replay at realistic scale	Tests the Section 6.2 alarm-flood controls
Coverage and cost	Tags monitored, readings processed per unit time, compute and storage consumed	Scale replay	Tests the Section 8.2 feasibility concern

Table 4. Evaluation design. Detection latency is identified as the primary metric because the framework's claimed benefit is earlier detection specifically, not detection as such: threshold monitoring eventually detects the compressor failure of Section 3.1, in week seven, which is the wrong week.

7.2 What the framework would produce once running

Once implemented, the pipeline would produce as natural output a count of tags monitored and readings processed per unit time, a per-event record of which check triggered each flag, and the flag history per tag that

supports instrument health tracking. These become the measurement substrate. They are described here as design outputs, not as results.

8. Limitations

8.1 Faults the framework cannot detect

The framework is built on statistical and rule-based comparison against a tag's own history and physically plausible bounds. It therefore cannot detect a fault that produces values statistically indistinguishable from healthy behaviour.

Three cases are outside its reach by construction. A sensor that fails in a manner that happens to track the correct physical trend produces no detectable inconsistency. A slow correlated drift affecting several instruments through a shared root cause, a common calibration reference or an environmental factor acting on all of them alike, defeats the cross-correlation test of Section 5.1, because nothing looks inconsistent relative to its neighbours. And errors introduced upstream of the sensor, incorrect placement, wrong instrument specification for the application, or human error in tagging and configuration, produce data that is internally consistent and wrong from the outset.

The third case deserves emphasis because it is the one most likely to be mistaken for a framework failure. A correctly functioning instrument measuring the wrong thing is not a data quality problem the data can reveal.

8.2 Cost and scale

Maintaining rolling baselines, drift detection, and cross-correlation per tag, continuously, across potentially tens of thousands of tags on a large site, is computationally non-trivial. Per-tag baseline storage and compute grow with tag count and history length. Cross-correlation scales worse, since the candidate relationship set grows faster than the tag count.

At genuine industrial scale this forces two design consequences that a naive implementation would miss: streaming computation rather than repeated batch recomputation over history, and deliberate scoping of which tags are checked against which neighbours rather than an all-pairs comparison. This is a real engineering constraint and is stated rather than deferred.

8.3 Residual risk when a fault passes

If a fault is not caught it propagates exactly as unvalidated telemetry does. Dashboards render it as normal, models treat it as legitimate signal, and decisions inherit the error with nothing to prompt a question.

The framework does not eliminate this. It reduces the surface on which it can happen silently, and the duration for which it can persist. A partially effective validation layer catches a meaningful share of common failure modes, not all of them, and it should be represented that way rather than as an assurance.

8.4 Dependencies

Baseline-derived thresholds require sufficient history per tag, so newly commissioned instruments and new equipment have limited protection during the period in which a baseline accumulates. Plausibility bounds and correlation relationships must be defined correctly per site and maintained as equipment changes, which requires domain input. The framework is therefore not a drop-in capability independent of site-specific engineering effort.

Most importantly, because it has not been tested against real operational data, its detection and false-alarm behaviour in a live environment is unknown rather than merely unproven. That distinction is deliberate.

8.5 What would be built next

The clearest next step is the evaluation of Section 7, since it is the binding gap. Beyond that, the most valuable extension is root-cause-aware flagging: distinguishing an isolated instrument problem from a shared upstream cause affecting several instruments together, which Section 8.1 identifies as something the current design explicitly cannot separate.

9. Conclusion

The argument of this paper is that sensor data quality should be enforced as a gate rather than repaired as a cleanup, and that the position of the check determines what it is capable of catching.

Three failure modes motivate it, and they share the property of being quiet. A faulty reading can be physically impossible and still be recorded. A drifting instrument stays inside every alarm limit by construction, because drift is a property of a sequence and a threshold is a test on a value. A frozen stream presents a plausible number at the expected cadence and is read as stability. In each case the systems that would ordinarily raise an alarm are the systems consuming the compromised data, so the absence of an alarm is not evidence of health.

The framework's response is to ask a different question before the data is used: not whether this value crossed a limit, but whether this value and this stream's recent behaviour remain consistent with a functioning instrument. Two discriminations make that operationally credible. Corroboration across physically coupled tags separates an instrument fault from a real process change, since physics propagates a genuine event and does not propagate a sensor failure. Persistence separates gradual drift from abrupt fault, since the two differ in shape rather than in magnitude. A graded quality status, rather than a binary verdict, lets a trend display and a model retraining pipeline treat the same imperfect reading differently, which a global accept or reject cannot do.

What this paper does not do is establish that the framework works. It has not been implemented, it has not been run against operational or historical data, and no detection rate, false-alarm rate, or latency figure exists for it. Those absences are stated rather than filled, and Section 7 sets out the evaluation, synthetic fault injection for controlled detection characteristics and historical replay for realism, that would be required to produce them. Detection latency is the metric that matters most, because threshold monitoring does eventually detect the failure in the worked example. It detects it in week seven.

References

- [1] R. Y. Wang and D. M. Strong, "Beyond Accuracy: What Data Quality Means to Data Consumers," *Journal of Management Information Systems*, vol. 12, no. 4, pp. 5-33, 1996. doi: 10.1080/07421222.1996.11518099
- [2] H. Y. Teh, A. W. Kempa-Liehr, and K. I. K. Wang, "Sensor data quality: a systematic review," *Journal of Big Data*, vol. 7, no. 11, 2020. doi: 10.1186/s40537-020-0285-1
- [3] R. Isermann, "Model-based fault-detection and diagnosis, status and applications," *Annual Reviews in Control*, vol. 29, no. 1, pp. 71-85, 2005. doi: 10.1016/j.arcontrol.2004.12.002
- [4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1-58, 2009. doi: 10.1145/1541880.1541882
- [5] E. S. Page, "Continuous Inspection Schemes," *Biometrika*, vol. 41, no. 1-2, pp. 100-115, 1954. doi: 10.1093/biomet/41.1-2.100
- [6] S. W. Roberts, "Control Chart Tests Based on Geometric Moving Averages," *Technometrics*, vol. 1, no. 3, pp. 239-250, 1959. doi: 10.1080/00401706.1959.10489860
- [7] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1-37, 2014. doi: 10.1145/2523813
- [8] J. Buggana, "Anomaly Detection in Industrial Pumps Using Streaming Sensor Data," *International Journal of AI, BigData, Computational and Management Studies*, vol. 7, no. 2, pp. 248-261, 2026. doi: 10.63282/3050-9416.ijaibdcms-v7i2p133
- [9] NAMUR, "NE 107: Self-Monitoring and Diagnosis of Field Devices," NAMUR Recommendation, 2006.
- [10] International Electrotechnical Commission, "IEC 62264, Enterprise-control system integration," IEC, Geneva.
- [11] International Electrotechnical Commission, "IEC 62541, OPC Unified Architecture," IEC, Geneva.
- [12] International Organization for Standardization, "ISO 8000, Data quality," ISO, Geneva.
- [13] Geetanjali, & Singh, A. R. (2025). Precision and smart farming: An approach to efficient resource management in agriculture. *Journal of Artificial Intelligence in Agriculture and Applied Research*, 1(1), 25-33. <https://agrpjournals.com/journals/journal/index.php/jaiaar/article/view/6>