International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 10 Issue 10, October - 2026, Pages: 1-19

Integrating Performance Engineering into Continuous Integration and Continuous Delivery Pipelines

Olumide Kumuyi[1*], Toussida Fatah Tanguy Minoungou[2], Aisha I. Abdulwahab[3], Serif Oyindamola Oyesiji[4], Samuel Fadero[5]

[1] New York University Abu Dhabi, Abu Dhabi, United Arab Emirates
[2] ThousandEyes (Cisco), USA
[3] Cummins Inc., Indiana, USA
[4] Harrisburg University of Science and Technology, Harrisburg, Pennsylvania, USA
[5] SanlamAllianz Life Insurance Ltd, Lagos, Nigeria
[*] *Corresponding author: Olumide Kumuyi, olukumuyi@gmail.com*

Abstract*: Purpose: Performance regressions are among the defects that continuous integration and delivery pipelines catch least reliably, because latency and throughput are distributions measured on noisy, shared infrastructure rather than pass or fail verdicts. This review asks how performance evidence should be generated, compared and acted upon inside a pipeline so that genuine regressions are stopped before release while false alarms and pipeline delay stay bounded. Method: A critical integrative review of software testing, performance engineering, statistical process monitoring and release governance literature, appraised for disclosure of the measurement environment, repetition, statistical treatment and transfer from build runners to production. Findings: Fixed single-run thresholds fail in both directions. Run-to-run variation on shared runners often exceeds the regressions that matter, so thresholds either flap or are loosened until real change slips through. The evidence converges on paired, interleaved measurement; comparison of distributions and tail percentiles through effect sizes with confidence intervals; change-point detection over build histories; and canary analysis that compares like with like. Contribution: The Noise-Aware Performance Gating (NAPG) framework links workload fidelity, noise control, change-point sensitivity, diagnostic observability and release governance through a signal detection mechanism, and states six falsifiable propositions with a stepped-wedge validation design. Implications: Performance gates should be engineered and audited as calibrated detectors with explicit error rates, not maintained as static latency limits.*

Keywords: performance regression detection, continuous delivery, change-point detection, tail latency, benchmark noise, canary analysis, observability

1 Introduction

Continuous integration and continuous delivery have shortened the interval between a code change and its exposure to users from weeks to hours. Functional correctness has largely kept pace with that compression, because unit, integration and acceptance tests return deterministic verdicts that a pipeline can act on without human judgment. Performance has not kept pace. A change that adds four milliseconds to a hot path, allocates an extra object per request, or introduces a lock around a shared cache passes every functional test and may still push the 99th percentile latency of a service beyond its service-level objective once production concurrency is reached. Such regressions are rarely noticed by the developer who introduces them, accumulate across many small commits, and are expensive to attribute after the fact, because by the time they surface in production dozens of unrelated changes have also been deployed (Oyesiji et al., 2026).

The difficulty is statistical before it is organisational. Latency is a distribution rather than a scalar, and its tail is shaped by queueing, garbage-collection pauses, cache misses and lock contention that appear only under concurrency. Pipelines, however, usually execute on shared cloud runners whose processor allocation, memory bandwidth, disk and network throughput vary with co-tenant load, so the same binary measured twice can differ by more than the regression of interest. A gate that compares one run of a candidate build against a fixed response-time limit therefore mixes three signals: the effect of the change, the variability of the environment and the representativeness of the workload. Treating the output as a verdict invites the failure pair familiar from industrial alarm systems, namely frequent false alarms that teach engineers to rerun or ignore the gate, followed by loosened limits that let real regressions through (Ekelemu, 2025a; Gideon et al., 2019).

Existing research addresses parts of this problem in isolation. Software testing texts and standards treat performance testing as a distinct, often late, activity whose load profiles derive from non-functional requirements. Regression testing research concentrates on the selection and prioritisation of functional tests under time budgets. Monitoring work concentrates on production telemetry and dashboards (Ogbole et al., 2021), and statistical change detection has matured in finance, public health and industrial condition monitoring without being systematically translated to build histories (Atakpa et al., 2024; Komi et al., 2021). What is missing is an integrated account of how workload design, measurement noise, statistical decision rules, diagnostic telemetry and release authority jointly determine whether a performance gate produces a credible signal, and of the conditions under which it cannot.

Practitioner guidance tends to prescribe tools, whereas academic studies tend to evaluate one detector on one benchmark suite; neither specifies the error rates a gate is expected to achieve or the evidence that would show it has failed.

The review is organised around three research questions. RQ1 asks which workload and environment conditions determine whether performance measurements taken inside a pipeline are valid proxies for production behaviour. RQ2 asks which statistical procedures detect latency and throughput regressions in noisy build histories with acceptable sensitivity and false-alarm rates, and at what cost in repetitions and pipeline time. RQ3 asks how percentile-based budgets, canary analysis and observability data should be combined into release decisions that engineers trust and act on. The questions are sequential: a statistical rule cannot rescue an invalid workload, and a well-calibrated detector adds little if its alarms are overridden without record.

This article makes four contributions. First, it recasts the performance gate as a detector in the sense of signal detection theory, which turns sensitivity, false-alarm rate and decision criterion into explicit design variables rather than side effects of an arbitrary limit. Second, it synthesises evidence on workload modelling, cloud benchmark noise, change-point detection, percentile estimation and canary analysis, and adjudicates where findings conflict, most notably on whether shared runners can support reliable gating at all. Third, it proposes the Noise-Aware Performance Gating (NAPG) framework, which connects five constructs to measurable pipeline outcomes and yields six falsifiable propositions, each with a stated test. Fourth, it specifies a stepped-wedge validation design with named threats to validity, so that the framework can be refuted as well as supported (Adebayo et al., 2026).

The scope is deliberately narrow. Continuous testing in general, including functional test automation, test flakiness and agile quality practice, is treated only where it bears on performance evidence. The unit of analysis is the gated change: a commit or merge request, the benchmark and load-test runs it triggers, the statistical comparison against a baseline, the release decision that follows, and the production behaviour that later confirms or contradicts that decision. Keeping this chain traceable, from change to measurement to decision to outcome, is itself a requirement of the framework, and it draws directly on the traceability literature. Section 2 sets out the theoretical lenses, Section 3 the review protocol, Section 4 the evidence synthesis, Section 5 the framework and propositions, Section 6 the discussion, Section 7 the limitations and research agenda, and Section 8 the conclusion.

2 Theoretical Foundations

Three bodies of theory make the problem tractable. Signal detection theory explains why a gate tuned for sensitivity alone will fail operationally and identifies the quantities that must be engineered instead. Queueing theory explains why the tail of the latency distribution, not its mean, is the quantity at risk, and why small service-time changes can have large consequences near saturation. Sociotechnical accounts of alarm systems explain why the same statistical detector can be trusted in one team and bypassed in another. Each lens supplies a different element of the framework developed in Section 5, and together they prevent the common error of evaluating a gate only by whether it caught a remembered incident (Arumosoye et al., 2021).

2.1 Signal Detection Theory and the Performance Gate

A performance gate observes a noisy quantity, such as the relative change in median or 99th percentile latency between a baseline build and a candidate build, and must decide whether that observation arose from noise alone or from noise plus a genuine regression. Signal detection theory formalises this situation. If the observed change X follows $N(0, \sigma^2)$ when there is no regression and $N(\delta, \sigma^2)$ under a regression of size δ, the discriminability of the detector is $d' = \delta/\sigma$, and the choice of decision criterion trades the hit rate against the false-alarm rate along a receiver operating characteristic curve. Two consequences follow. First, no threshold can compensate for low discriminability: when run-to-run variation σ is comparable to δ, every criterion yields either many false alarms or many misses. Second, d' rises only by reducing σ, through better noise control, more repetitions or paired measurement, or by increasing the effective δ, through workloads that exercise the changed code path under realistic concurrency. With n independent repetitions per side in a two-sample comparison, σ shrinks to approximately $\sigma_1\sqrt{(2/n)}$, where σ_1 is the single-run standard deviation, which is why the repetition budget is a central design decision rather than an implementation detail (Komi, 2023).

The optimal criterion also depends on costs and base rates. If regressions are rare, for example one change in fifty, even a detector with a 5 percent false-alarm rate per comparison produces more false alarms than true detections, and the positive predictive value of an alarm falls below one half. In a pipeline that evaluates hundreds of benchmarks per build, the family-wise probability of at least one false alarm, $1 - (1 - \alpha)^m$, approaches certainty; with $m = 200$ benchmarks and $\alpha = 0.05$ it exceeds 0.99. The criterion should therefore be set against the expected cost of a missed regression (incident handling, capacity spend, user impact) and the cost of a false alarm (engineer time, a blocked release, eroded trust), and multiplicity must be controlled, for instance through false discovery rate procedures such as Benjamini-Hochberg. Laboratory diagnostics offers a close analogy: limits of detection, sensitivity and specificity are fixed before a test is allowed to drive decisions, and a positive result is interpreted against prevalence (Eze et al., 2024b).

2.2 Queueing Behaviour, Tail Latency and Percentile Estimation

Queueing theory explains why performance regressions are disproportionately visible in the tail. For a single-server queue with Poisson arrivals and exponential service times, mean response time is $R = S/(1 - U)$, where S is the mean service time and $U = \lambda S$ is utilisation. A 5 percent increase in S at 70 percent utilisation raises U to 0.735 and R by roughly 19 percent. Because response time in this model is exponentially distributed with rate $(1 - U)/S$, the p-th percentile equals $-\ln(1 - p) \cdot S/(1 - U)$, so every percentile inherits the same amplification, and in real systems with heavier tails the high percentiles usually move further than the mean. Little's Law, $L = \lambda W$, links the same change to queue length and therefore to memory, thread-pool and connection-pool pressure. Real services are not single-server queues, but the qualitative lesson generalises: service-time regressions are amplified near saturation, and a benchmark run at low utilisation will understate the production consequence of a change (Adelanwa et al., 2023a).

Percentiles bring their own measurement hazards. Estimating the 99th percentile needs enough observations in the tail: with 1,000 requests only about 10 lie above the true p99, so the estimate has wide sampling error, and with 100 requests the reported p99 is essentially the maximum. Percentiles cannot be averaged across hosts or time windows; the mean of per-host p99 values is not the fleet p99, and aggregation must merge the underlying histograms or mergeable sketches such as HDR histograms, t-digest or DDSketch. Load generators that wait for each response before sending the next request suffer from coordinated omission: when the system stalls the generator stops sending, the stall is recorded once rather than for every request that would have arrived during it, and the reported tail becomes optimistic. Open workload models, in which arrivals are independent of completions, avoid this bias, whereas closed models with think time represent interactive users faithfully but understate queueing under overload. Choosing the workload model therefore determines which regressions the gate is able to see (Ogundairo et al., 2023b).

2.3 Alarm Fatigue, Trust and Decision Rights

A statistically sound gate can still fail if engineers do not act on it. Studies of alarm management in continuous process industries show that standing, nuisance and chattering alarms condition operators to acknowledge without diagnosing, and that alarm rationalisation, which assigns every alarm a priority, a defined response and an owner, restores their meaning; they also show that information lost at shift handover undermines follow-through on alarms that were acknowledged (Ekelemu, 2025a). Performance gates exhibit the same dynamics. A gate that fails intermittently for reasons unrelated to the change is retried until it passes, which converts it into a random delay and teaches the organisation that a red performance stage is noise. Trust is calibrated by experience: a detector with a known and published false-alarm rate, whose alarms arrive with diagnostic evidence, is more likely to be heeded than one whose failures must be investigated from scratch. Research on user trust in automated remote monitoring reaches a similar conclusion about transparency of the alerting logic (Akintola et al., 2025).

The sociotechnical lens also locates authority. A gate that can block a release moves decision rights from the author of the change to whoever owns the budget and the detector. When that ownership is ambiguous, blocked changes are overridden informally and the override leaves no trace, which destroys the feedback needed to recalibrate the gate. Governance research from other settings, covering key performance indicator systems, continuous control monitoring and incentive design, converges on the same requirement: every indicator needs an owner, an explicit threshold, a defined response and a periodic review of whether the threshold still reflects risk, and the parties who bear the cost of a control must see its benefit (Sakyi et al., 2022b; Adebayo et al., 2025).

3 Review Protocol

The review follows the logic of a critical integrative review. It combines controlled studies, industrial experience reports, standards and conceptual work in order to build and qualify a mechanism, rather than to pool effect sizes. Procedures were adapted from established guidance for systematic reviews in software engineering and from current reporting standards for evidence synthesis, with one deliberate difference: appraisal weighs each source by its relevance to the mechanism as well as by the quality of its design. Table 1 summarises the protocol, and Figure 1 shows the sequence of stages, including the route by which null and negative findings re-enter screening. The protocol was designed so that the framework in Section 5 could be traced back, construct by construct, to the kinds of evidence that support it and to the kinds that remain missing.

3.1 Search Strategy and Eligibility

Searches covered IEEE Xplore, the ACM Digital Library, Scopus, Web of Science and SpringerLink for works published between January 1994 and June 2026, with the lower bound chosen to capture foundational work on requirements traceability and test adequacy. The core string combined three concept blocks: (performance OR latency OR throughput OR benchmark* OR "load test*") AND ("continuous integration" OR "continuous delivery" OR "continuous deployment" OR pipeline OR DevOps OR "regression test*") AND (regression OR "change point" OR detect* OR noise OR variab* OR canary OR percentile OR observab*). Standards and bodies of knowledge were retrieved directly, and backward and forward snowballing from included

works located material that database indexing missed. Works were eligible if they reported or specified how performance, reliability or monitoring evidence is generated, compared or acted upon in an automated or continuously operating system. Works were excluded if they addressed functional correctness only, without a transferable measurement or decision component, if they lacked enough methodological detail to appraise, or if they were promotional material without method. Works on monitoring, anomaly detection, workforce learning and governance from other sectors were admitted only as mechanism analogies and were never used as evidence of effect sizes in software systems (Amarahobu et al., 2025a).

3.2 Appraisal, Extraction and Synthesis

Appraisal used five questions tailored to performance evidence: whether the measurement environment (hardware, virtualisation, runner type and isolation) was disclosed; whether the number of repetitions and the comparison statistic were reported; whether regressions had an independent ground truth, such as injected slowdowns or confirmed production incidents, rather than detector output alone; whether false alarms were counted alongside detections; and whether results were shown to transfer from the measurement environment to production. Extraction recorded the workload model, run count, test statistic, threshold or criterion, detection and false-alarm outcomes, pipeline time cost and the decision that followed. Synthesis was mechanism-oriented: for each construct, findings were compared across settings, contradictory results were retained and examined for moderating conditions, and each claim was graded by the strength of the evidence behind it. Standards were read as statements of expected controls, not as evidence that those controls improve outcomes. The review reports no pooled estimates and claims no exhaustive coverage; its product is a qualified mechanism and a set of propositions awaiting empirical test.

Table 1. Review protocol for evidence on performance regression detection in CI/CD pipelines

Protocol stage	Operational rule	Safeguard against bias
Sources	IEEE Xplore, ACM Digital Library, Scopus, Web of Science, SpringerLink; ISO/IEC/IEEE 29119-2 and 29148; SWEBOK v3.0	Database and citation routes combined to offset indexing gaps
Time window	January 1994 to June 2026	Foundational works retained; recency not treated as a proxy for quality
Search string	Performance terms AND pipeline terms AND detection terms (Section 3.1)	String piloted against known relevant works before the final run
Inclusion	Evidence on generating, comparing or acting on performance or monitoring data in automated or continuously operating systems	Criteria applied at full text; exclusion reasons logged
Exclusion	Functional-only testing; promotional material without method; insufficient methodological detail	Null and negative results explicitly retained
Appraisal	Environment disclosed; repetitions and statistic reported; independent regression ground truth; false alarms counted; transfer to production shown	Design quality and mechanism relevance weighted separately
Extraction	Workload model, run count, test statistic, criterion, detection and false-alarm outcomes, pipeline time cost, resulting decision	Fixed extraction form for each construct
Synthesis	Mechanism, context and outcome compared across five constructs	Conflicts adjudicated by moderating condition, not by vote counting
Cross-domain works	Admitted as mechanism analogies only	Never used as evidence of effect sizes in software systems

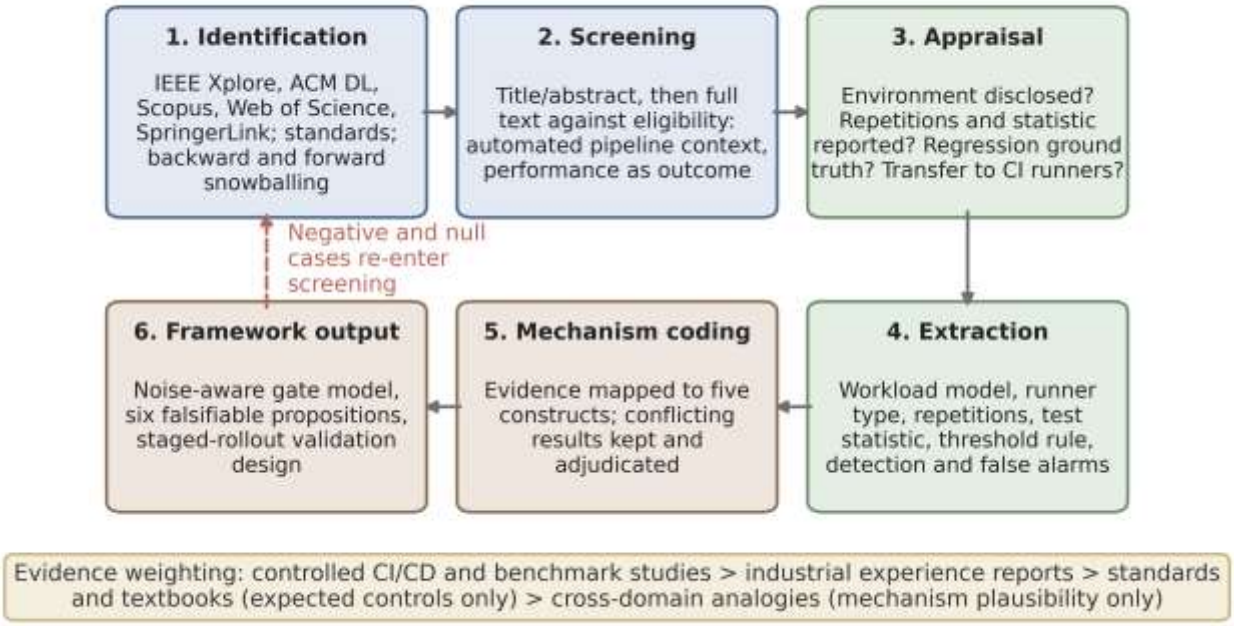


Figure 1. Review protocol stages from identification to framework output, with null and negative cases fed back into screening

Figure 1 makes one constraint visible. Inclusion in the evidence base does not by itself confer evidential weight; a source informs the framework only where it bears on a construct, a rival explanation, a boundary condition or a validation choice, and cross-domain sources are confined to the lowest tier of the weighting shown beneath the stages. The feedback arrow reflects a rule applied throughout: studies reporting that a detector failed, or that a gate was abandoned, were searched for as actively as studies reporting success (Fadayomi et al., 2019).

4 Evidence Synthesis

This section synthesises the evidence construct by construct. It moves from the conditions under which pipeline measurement is valid (workload fidelity and environmental noise), through the statistical decision rule, to the use of the result in budgets, canary releases and diagnosis. The final subsection adjudicates the principal conflicts in the literature rather than leaving them side by side. Throughout, the synthesis distinguishes what has been shown in controlled benchmarking or industrial settings from what follows analytically from measurement and queueing theory and from what remains conjecture. That distinction matters because much of the practitioner literature reports tools that worked in one organisation without reporting the noise regime, workload or decision rule under which they worked, which makes transfer claims impossible to evaluate (Oyesiji et al., 2026).

4.1 Workload Modelling and Fidelity

A workload model specifies the arrival process (open, closed or partly open), the request mix across endpoints or operations, the distribution of payload sizes and parameter values, the state of the data (database size, key skew and cache hit ratio), the level of concurrency, and the duration and warm-up of the run. Each element can make a pipeline measurement unrepresentative. A test database a hundredth the size of production is cache-resident, so a change that adds an index lookup costs nothing in the pipeline and a disk read in production. A uniform key distribution hides hot-key contention that Zipf-like production traffic exposes. A run that is shorter than the warm-up of a just-in-time compiler measures the interpreter. The testing literature has long framed adequacy in terms of which parts of the program and its input space a test exercises, and the same reasoning applies to load: a workload is adequate for a regression only if it drives the code path and data regime in which that regression lives (Eyetsemitan et al., 2023b).

Fidelity can be made measurable. If p is the distribution of requests across operations in production and q the corresponding distribution in the test workload, the Jensen-Shannon divergence JS(p, q), which lies between 0 and 1 in base 2, gives a single mix-fidelity score that can be tracked per release. Complementary indicators are the share of production processor time, taken from continuous profiling, that falls in functions the workload also exercises; the ratio of test to production data volume for each store; and the ratio of peak test concurrency to production p95 concurrency. Trace replay, in which sampled production traces are anonymised and replayed at controlled rates, raises fidelity but inherits the sampling bias of the tracer and can leak personal data unless redacted. Synthetic workloads derived from operational profiles are safer and easier to vary but drift from production as usage evolves. Inspection disciplines outside software face the same coverage problem: monitoring programmes for pipelines and transmission corridors state explicitly which segments are observed at what resolution, so that the absence of an alarm is not mistaken for the absence of a defect (Dagodzo et al., 2020).

A recurring trade-off separates microbenchmarks from system-level load tests. Microbenchmarks isolate a function, run in seconds, have low variance and can be executed on every commit, but they omit the interactions (queueing, contention, allocation pressure across components) through which many production regressions arise. System-level tests reproduce those interactions but take tens of minutes, demand more infrastructure and show higher variance. The evidence does not support choosing one over the other. It supports assigning them different roles: microbenchmarks as fast, low-cost leading indicators with advisory status, and system-level tests at production-like utilisation as the evidence on which blocking budgets rest (Obogo et al., 2023).

4.2 Benchmark Noise on Shared Cloud Runners

The variability of performance measurements on cloud-hosted continuous integration runners has several distinct sources. Co-tenant load competes for last-level cache, memory bandwidth and, on oversubscribed hosts, processor time (visible as steal time). Instances of the same nominal type may run on different processor generations, microcode versions or memory configurations. Dynamic frequency scaling and turbo behaviour depend on thermal and power state. Burstable instance types throttle once credits are exhausted, and network and block storage are rate-limited in ways that vary between runs. Inside the process, just-in-time compilation, garbage collection timing, memory layout and address-space randomisation can shift performance by several percent between otherwise identical runs, and code alignment effects mean that an unrelated change can move a benchmark. Distributions of repeated measurements are frequently skewed, heavy-tailed or bimodal rather than normal. Commonly reported coefficients of variation range from well below 1 percent on dedicated, tuned hardware to more than 10 percent for input-output-bound benchmarks on shared virtual machines. Measurement science in physical inspection offers the right framing: the capability of the instrument must be established, through repeatability and reproducibility studies, before its readings are compared with an acceptance limit (Atta et al., 2022).

It is useful to decompose the variance of a single measurement into a between-instance component and a within-instance component, $\sigma^2 = \sigma^2_between + \sigma^2_within$. Repeating a benchmark many times on one runner shrinks only the within-instance term; a comparison of a candidate measured today against a baseline stored from a different runner last week carries the full between-instance term and any drift since. Paired designs remove the between-instance term by measuring baseline and candidate on the same runner in the same job, ideally interleaved in randomised order (for example, ABBA or randomised multiple interleaved trials) so that slow drift within the job affects both builds equally. The required sample size follows from the power calculation for a two-sample comparison of log-latency: $n \approx 2(z_{1-\alpha/2} + z_{1-\beta})^2(CV/\delta)^2$ runs per side. With a coefficient of variation of 10 percent, a budget δ of 5 percent, $\alpha = 0.01$ and power 0.8, this gives about 94 runs per side; halving the effective coefficient of variation through pairing and isolation to 3 percent reduces the requirement to about 9. Noise control is therefore usually cheaper than repetition, a point that cost-governance analyses of technology spending would predict (Adesuyi et al., 2023).

Other mitigations trade validity for stability. Dedicated or bare-metal runners with fixed frequency, isolated cores and disabled turbo cut variance substantially but cost more and resemble production less. Hardware-in-the-loop validation pipelines for embedded and vehicle systems illustrate the same choice in physical form, since dedicated test benches are introduced precisely to obtain controlled, repeatable execution conditions that shared virtual runners cannot guarantee, at the cost of capacity and of distance from field conditions (Oyesiji et al., 2026). Hardware counters such as retired instructions, or simulation-based measures from cache simulators, are nearly deterministic and ideal for commit-stage microbenchmarks, yet they do not register effects that act through wall time, such as lock contention, input-output waits or changes in parallelism. A defensible configuration uses deterministic counters to flag suspicious commits cheaply and wall-time paired comparisons at production-like utilisation to decide. The appraisal found that studies rarely report A/A comparisons (the same build compared with itself), although these are the only direct measurement of a gate's false-alarm rate in its real environment (Gideon et al., 2019).

4.3 Statistical Detection: From Thresholds to Change Points

Statistical procedures for regression detection fall into two families, summarised in Table 2. Pairwise procedures compare a candidate build with a baseline. Welch's t-test on log-transformed latencies is efficient when run-level summaries are roughly log-

normal; the Mann-Whitney U test with a Hodges-Lehmann shift estimate is robust to outliers and asks whether the candidate is stochastically slower; bootstrap confidence intervals for the ratio of medians or of 99th percentiles give an effect size in the same units as the budget; and equivalence testing through two one-sided tests can pass a build affirmatively by showing that any change is smaller than the budget, instead of passing it merely because no difference was detected. History-based procedures treat the per-build summary as a time series. Statistical process control charts such as CUSUM and EWMA accumulate evidence of a persistent shift. Offline change-point methods such as E-Divisive means, which uses an energy statistic and makes no distributional assumption, and PELT, which minimises a penalised cost, locate the builds at which the distribution changed. Bayesian online change-point detection maintains a posterior distribution over the time since the last change (Atakpa et al., 2024).

Table 2. Statistical procedures for detecting performance regressions in pipelines

Procedure	Question answered	Key assumption	Main strength	Main weakness or cost
Fixed limit on a single run	Did this run exceed a limit?	Noise negligible relative to the limit	Trivial to implement and explain	False alarms and misses whenever noise is comparable to the budget
Welch t-test on log latency	Did the mean shift?	Run summaries roughly log-normal and independent	Efficient at modest run counts	Sensitive to outliers and bimodal runs
Mann-Whitney U with Hodges-Lehmann shift	Is the candidate stochastically slower?	Independent runs of similar shape	Robust to outliers; yields a shift estimate	Weak for changes confined to the tail
Bootstrap interval for a percentile ratio	How large is the change in p50 or p99?	Runs exchangeable within a build	Effect size in the units of the budget	Needs many requests for a stable tail estimate
Equivalence test (two one-sided tests)	Is any change smaller than the budget?	Budget fixed in advance	Passes a build on positive evidence	Needs more runs than a difference test
CUSUM or EWMA chart	Has the build series drifted?	Known in-control mean and variance	Fast detection of small persistent shifts	Must be re-baselined after accepted changes
E-Divisive or PELT change points	Where in the history did the distribution change?	Piecewise-stationary series	Localises the commit range; catches creeping regressions	Retrospective; penalty choice drives false positives
Bayesian online change point	How probable is a change at this build?	Specified hazard rate and likelihood	Probabilistic output for cost-weighted decisions	Misspecification under heavy tails

The CUSUM statistic illustrates the logic of history-based detection. With standardised observations $z_t = (x_t - \mu_0)/\sigma_0$, the upper CUSUM is $S_t = \max(0, S_{t-1} + z_t - k)$, and an alarm is raised when S_t exceeds h. With $k = 0.5$ and $h = 5$, the in-control average run length is about 465 builds and a sustained shift of one standard deviation is detected after about 10 builds on average. These are the parameters a team should reason about: how many builds between false alarms it can tolerate, and how many builds of delay it will accept before a regression is flagged. The history-based family has two distinctive strengths. It detects creeping regressions made of steps that are individually below the pairwise budget, which a build-to-build comparison never sees, and it localises the change to a range of commits, which bounds the subsequent bisection. Its weaknesses are equally specific: offline methods are retrospective and need builds after the change to confirm it, penalty and threshold choices drive the false-positive rate, and every accepted change requires re-baselining (Komi et al., 2021).

Figure 2 illustrates the contrast with a simulated history of 120 builds. The p99 latency carries multiplicative noise of about 4.5 percent, seven builds are inflated by 10 to 18 percent to mimic noisy-neighbour interference, and a true 6 percent regression begins at build 70. A fixed limit at 8 percent above the baseline raises six alarms before the regression, all false, and then flaps, firing on 20 of the 50 regressed builds, so neither its alarms nor its passes carry information. A single change-point split on the log series places the change at build 71, and a CUSUM chart calibrated on the first 30 builds signals at build 77. Panel (b) shows the analytic power of a paired two-sample comparison for a coefficient of variation of 10 percent at $\alpha = 0.01$: detecting a 5 percent regression

with probability 0.8 needs far more repetitions than most pipelines schedule, which is why reducing variance matters more than choosing a cleverer test (Obogo et al., 2026).

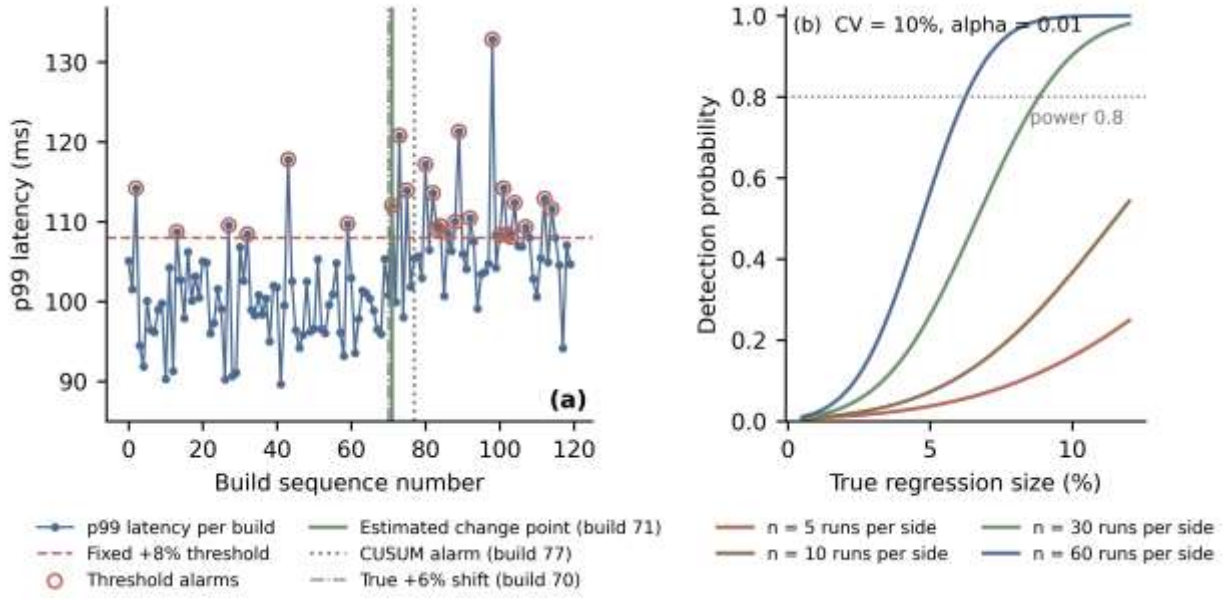


Figure 2. Illustrative simulated build history: (a) a fixed limit produces false alarms and flapping while change-point methods locate a 6 percent regression; (b) analytic detection probability by regression size and runs per side

Two further issues cut across both families. The first is multiplicity: a suite of hundreds of benchmarks, each tested at conventional significance, will fail some build every day by chance. Controlling the false discovery rate across benchmarks, and deduplicating alarms that share a root cause (several benchmarks that call the same regressed function), keeps the alarm stream interpretable; similarity-based de-duplication of alerts, of the kind developed for shared threat indicators, is a promising technique for the second task (Adebayo et al., 2026). The second is the choice of statistic. Median-based tests are stable but insensitive to regressions confined to the tail, such as a new slow path taken by 2 percent of requests; tail-based comparisons are sensitive to these but need many more requests per run. A gate that protects a p99 objective must therefore test the p99, or a distributional statistic such as the Kolmogorov-Smirnov distance, and budget the request volume that a stable tail estimate requires.

4.4 Latency Percentiles, Performance Budgets and Service-Level Objectives

A performance budget converts a service-level expectation into a limit that a single change may consume. Budgets are absolute (p99 at or below 250 milliseconds at 500 requests per second on the reference workload) or relative (no more than a 3 percent increase in median and 5 percent in p99 against the baseline). Resource budgets complement latency budgets: processor time per request, allocations per request, peak memory, cold-start time and artefact size are cheaper to measure, less noisy and often earlier indicators of a latency regression. The choice of percentile should follow the architecture. In a request that fans out to 100 backend calls and waits for all of them, the probability that at least one call falls in the backend's slowest 1 percent is 1 - 0.99^100, or about 63 percent, so backend p99 becomes the user's typical experience; fan-out services need budgets on p99 or p99.9, whereas batch and throughput-oriented services are better served by budgets on median cost (Aminu-Ibrahim et al., 2025c).

Budgets connect naturally to service-level objectives and error budgets in production. An objective such as 99.9 percent of requests completing within 300 milliseconds over 28 days leaves an error budget of 0.1 percent of requests. The burn rate is the ratio of the observed fraction of bad requests to that allowance, and multi-window burn-rate alerting pages only when the budget is being consumed quickly over both a short and a long window; a burn rate of 14.4 sustained for one hour, for example, consumes 2 percent of a 30-day budget. A pipeline budget is, in effect, a forecast of how much of the production error budget a change will consume. Treating the two as one accounting system lets teams ask whether a release that uses 1 percent of latency headroom is worth what it delivers, which is the question performance indicators in other sectors are designed to support (Sakyi et al., 2022b). Service-assurance frameworks for enterprise IT systems that support large user populations draw the same connection, treating reliability and responsiveness objectives as operational commitments that every change must be shown to respect before it reaches users.

A subtle but important design choice is the reference point. If each build is compared with its immediate predecessor, a sequence of changes each consuming 2 percent of a 3 percent budget passes every gate while latency drifts upward without limit, a ratchet that history-based detection can catch but pairwise gates cannot. Anchoring budgets to a fixed reference release, re-baselined only by an explicit, owned decision, prevents this drift and makes the consumption of headroom visible. Governance frameworks for long-lived assets make the same argument for measuring against a fixed design baseline rather than the last inspection (Aminu-Ibrahim et al., 2024).

4.5 Canary Analysis and Progressive Delivery

Canary analysis moves the comparison into production. A new version receives a small share of real traffic, and its metrics are compared with those of the current version before exposure widens. The evidence identifies several design choices that decide whether canary verdicts are credible. The comparison should be between the canary and a baseline deployed at the same moment on fresh hosts of the same size and receiving the same traffic share, not between the canary and the long-running fleet, which differs in cache warmth, uptime, memory fragmentation and connection state. Metrics should be classified in advance by direction and importance, each compared with a robust test such as Mann-Whitney U, and combined into a score whose pass mark was fixed before the release. Open-source automated canary analysis tools follow this pattern, and it guards against the temptation to explain away a bad metric after the fact. A composite score is only as good as its components and weights; the medical literature on composite risk indices shows that a single number summarising several markers must be validated against the outcome it is meant to predict (Eze et al., 2024b).

Canaries face a power problem that mirrors the benchmark noise problem. A 1 percent traffic slice over 30 minutes on a service handling 200 requests per second yields about 3,600 canary requests, enough for a stable median but only about 36 observations above the p99. Evaluating the results repeatedly as data accumulate, and stopping as soon as they look acceptable, inflates the false-pass rate; sequential procedures such as the sequential probability ratio test or always-valid confidence sequences allow continuous monitoring without that inflation, and fixed evaluation horizons are the simpler alternative. Canaries also have characteristic blind spots: effects that depend on scale (shared-cache hit rates, database contention and connection limits), effects that accumulate over time (memory leaks and compaction debt), and effects confined to tenants or regions that the canary did not receive. Shadow traffic, in which requests are duplicated to the new version without serving its responses, removes user exposure but not the scale problem (Ilodigwe et al., 2024a).

The operational value of a canary lies as much in the readiness to reverse as in the analysis. Rollback must be rehearsed, fast and automated, and the conditions that trigger it must be defined before the release, much as emergency response capability in process facilities is judged by drills rather than by plans (Aminu-Ibrahim et al., 2018). Performance optimisation frameworks for high-demand enterprise technology rollouts make the same point at programme level, pairing staged exposure and capacity headroom with a prepared path back so that a degraded release can be withdrawn before demand peaks (Olagunju et al., 2024).

4.6 Observability for Diagnosis and Attribution

Detection without diagnosis shifts cost rather than removing it. An alarm that says "p99 is 7 percent higher" leaves an engineer to reproduce the result, identify the responsible commit and locate the code path, which in a busy repository can take longer than the regression is worth. The observability literature distinguishes metrics, logs and traces, now commonly unified through OpenTelemetry instrumentation, and adds continuous profiling as a fourth signal. For services, the RED method (rate, errors, duration) summarises user-facing behaviour; for resources, the USE method (utilisation, saturation, errors) locates the bottleneck. Differential flame graphs, which subtract the stack-sampled profile of the baseline from that of the candidate, point directly at the functions whose cost changed. Exemplars link a histogram bucket, such as the slowest 1 percent of requests, to specific traces, so that a tail regression can be inspected request by request (Ogundairo et al., 2023b; Ogbole et al., 2021).

Attribution to a commit is the second diagnostic task. Automated bisection is attractive but fragile when the predicate is noisy: a bisection step that wrongly classifies one build sends the search down the wrong half of the range. A noise-aware predicate, which repeats and pairs measurements at each step and abstains when the confidence interval straddles the budget, is slower per step but converges on the right commit. History-based change-point detection helps by narrowing the initial range. Telemetry has costs that bear on the design: high-cardinality labels inflate metric storage, full tracing is expensive, and head-based sampling decides before a request's latency is known, so tail-based sampling, which retains slow and failed traces, is preferable for diagnosing tail regressions. Traces and profiles may contain personal data and must be governed accordingly. Operational monitoring in digital-twin and substation settings, and interpretable failure prediction in distributed energy assets, show the same principle: a detector earns its place when its output can be explained and traced to a cause (Komi et al., 2021).

4.7 Adjudicating Conflicting Evidence

Four disagreements recur in the literature, and each can be resolved by identifying the condition that separates the opposing findings rather than by counting studies on each side. The first concerns whether shared cloud runners can support performance gating at all. Pessimistic accounts report that run-to-run variation on shared virtual machines exceeds the regressions of interest, while optimistic accounts report reliable detection of changes of a few percent. The designs differ: pessimistic results typically compare against stored baselines or use few repetitions, whereas optimistic results use paired, interleaved measurement on the same instance with many repetitions. The adjudication is that shared runners support gating for regressions larger than roughly three times the paired standard error, and do not support comparisons against historical baselines from other instances. The disagreement is about design, not about the cloud (Sanni et al., 2022).

The second disagreement concerns microbenchmarks. Some studies find that microbenchmark changes predict little about production performance; others find that they catch regressions early and cheaply. Both findings hold because the two kinds of test measure different constructs; the resolution, anticipated in Section 4.1, is to assign them different decision rights. The third concerns where testing belongs. Advocates of production-first practice argue that no pre-production workload is faithful and that canaries and objectives are the only trustworthy evidence; critics point to canary blind spots, to user exposure and to the cost of learning from incidents. The error profiles are complementary: pre-production gates catch regressions that are large relative to benchmark noise and exercised by the workload, while canaries catch regressions that depend on real traffic mix, so the defensible design is layered (Oluwo et al., 2025).

The fourth disagreement is between statistical gates and simple thresholds. Engineers report that statistical gates are opaque and simple limits easy to understand, while statisticians point out that simple limits have unknown error rates. The resolution lies in presentation rather than method: effect sizes with confidence intervals in milliseconds and percent, set against the budget, are as easy to read as a threshold and carry their own uncertainty, whereas p-values without effect sizes are neither. Across all four disputes, the lesson is that performance gating evidence is conditional on design, and that claims about what works must specify the workload, the noise regime and the statistic.

5 The Noise-Aware Performance Gating Framework

The Noise-Aware Performance Gating (NAPG) framework integrates the synthesis into an explanatory model. Its central mechanism is that a performance gate improves release outcomes only when it produces a credible signal, meaning a regression signal with high discriminability, a bounded and known false-alarm rate, and attribution to a specific change, and when that signal is acted upon through owned decision rules. Five constructs determine whether the mechanism operates. Workload fidelity (C1) and noise control (C2) set the achievable discriminability; change-point sensitivity (C3) sets how that discriminability is converted into decisions; diagnostic observability (C4) determines whether alarms can be acted on quickly; and release governance (C5) moderates whether they are acted on at all. The framework is an application of design science reasoning: it specifies an artefact, the conditions of its use and the evidence that would show it works, and it remains a proposal until that evidence exists (Eze et al., 2022b).

5.1 Constructs and Mechanism

Figure 3 shows the structure of the framework. The first three constructs feed the credible-signal mechanism directly, because together they determine d' and the criterion. Diagnostic observability enters the mechanism at the point of attribution, turning a detected shift into a located cause. Release governance acts as a moderator: the same signal leads to a blocked release, a recorded override or a silent retry depending on who owns the budget and what response has been defined. Outcomes are measured on five dimensions at once, namely regressions caught before release, false alarms per 100 gated builds, time from alarm to attributed cause, pipeline delay added per change, and production tail latency together with objective burn. The dimensions are measured jointly because improvements trade off: raising repetitions improves detection and adds delay, and tightening the criterion reduces escapes and raises false alarms. Boundary conditions (heterogeneous runners, drift in traffic mix, caching and asynchronous work, and low canary traffic) weaken the link between constructs and signal, and the feedback path carries escaped regressions and false alarms back into the workload model, baselines and budgets, in the manner of organisational learning loops described for continuous safety improvement (Arumosoye et al., 2021).

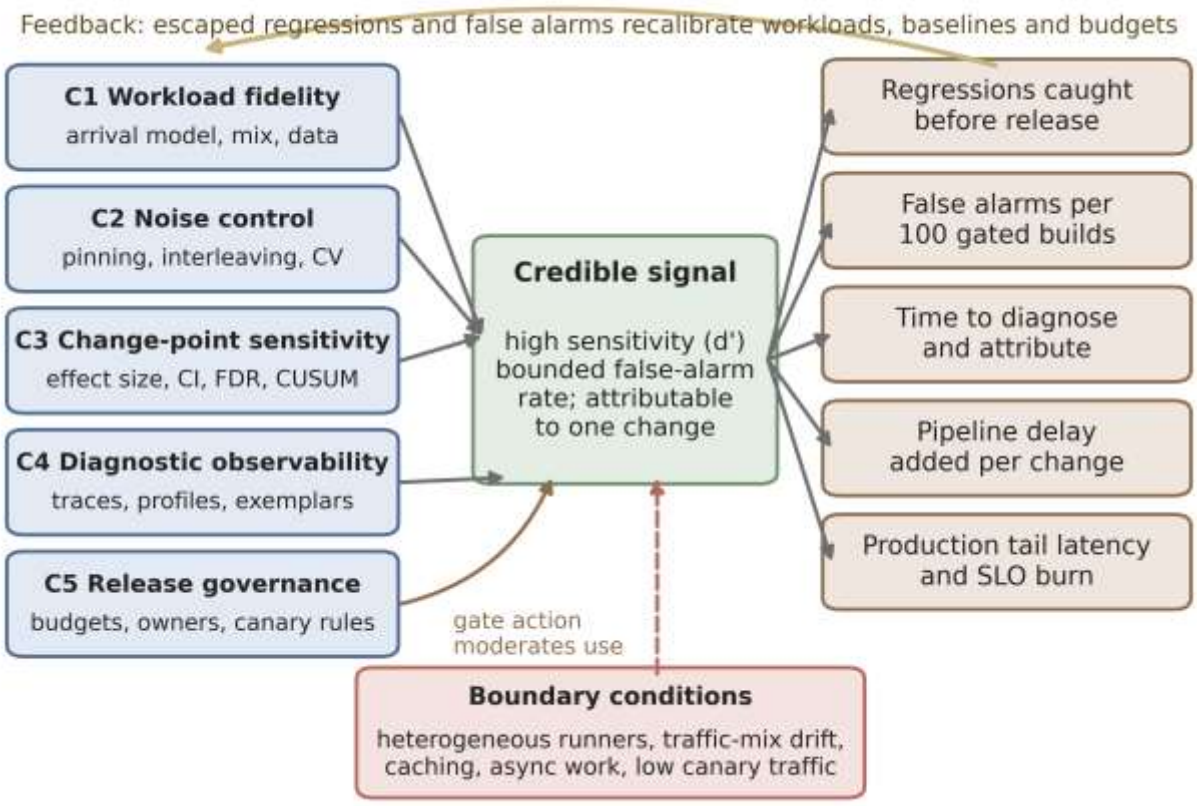


Figure 3. The Noise-Aware Performance Gating (NAPG) framework: constructs, credible-signal mechanism, outcomes, boundary conditions and feedback

The mechanism implies an ordering of investment. Where noise is high relative to budgets, changing the statistical method achieves little until noise control improves; where noise is low but workloads miss the paths in which regressions occur, repetitions add cost without reducing escapes; and where detection is good but alarms are overridden without record, better diagnosis and clearer ownership pay off first. The framework therefore functions as a diagnostic for the gate itself, directing attention to whichever construct is currently the binding constraint, much as maturity models in engineering organisations identify the capability whose absence limits the rest (Abdulhadi et al., 2026b).

5.2 Operational Indicators

Each construct is measured by indicators with explicit formulas, failure signals and prescribed gate actions, as set out in Table 3. The indicators are chosen so that they can be computed from data the pipeline already produces, together with two additions most pipelines lack: scheduled A/A comparisons, which measure the false-alarm rate directly, and injected-slowdown trials, in which a known delay is inserted into a build to measure detection rate and delay against ground truth. The failure signals are observable without a controlled study, which allows teams to audit their own gates, and every signal maps to a defined action so that the indicator changes behaviour rather than simply reporting it (Aliliele et al., 2024a).

Table 3. NAPG constructs, indicators, failure signals and gate actions

Construct	Definition	Indicator	Failure signal	Gate action
C1 Workload fidelity	Degree to which the test load reproduces production arrivals, request mix and data state	Jensen-Shannon divergence of operation mix; share of production CPU time in exercised paths; test to production data-volume ratio	Escaped regressions in paths the workload never exercises	Refresh workload from recent traces; add a scenario for each escaped path

Construct	Definition	Indicator	Failure signal	Gate action
C2 Noise control	Size of run-to-run variation relative to the budget	Paired A/A coefficient of variation; between-instance variance share; A/A false-alarm rate	A/A comparisons fail more often than the nominal alpha	Adopt interleaved pairs, pin or isolate runners, or raise repetitions
C3 Change-point sensitivity	Probability that a budget-sized regression is flagged within k builds	Detection rate and median delay on injected slowdowns; false discovery rate across benchmarks	Injected slowdowns missed or flagged late	Retune criterion; add CUSUM on history; apply FDR control
C4 Diagnostic observability	Ability to attribute a flagged regression to a change and a code path	Median time from alarm to attributed cause; share of alarms with a differential profile attached	Alarms closed as could not reproduce	Attach profiles and traces automatically; use noise-aware bisection
C5 Release governance	Existence and use of owned budgets with defined responses	Share of budgets with owner and review date; override rate with logged reasons; escaped-regression rate	Silent overrides; budgets never revised	Name owner; log overrides; review budgets each quarter against a fixed anchor

5.3 Decision Flow Across Pipeline Stages

Figure 4 translates the framework into a decision flow. At the commit stage, deterministic counters and microbenchmarks flag suspicious changes cheaply and never block on their own. Scheduled or merge-time load tests run baseline and candidate interleaved on the same runner pool, and the statistical comparison produces a relative change with a bootstrap confidence interval, with false discovery rate control across metrics. The decision rule has three outcomes rather than two. If the upper confidence bound lies within budget, the change proceeds; if the lower bound lies above budget, the change is blocked and diagnosis starts automatically, with bisection, a differential flame graph and notification of the owner; if the interval straddles the budget, the result is inconclusive and more runs are added sequentially until it resolves or a maximum is reached. Changes that pass enter a canary stage with a pre-registered pass mark, and promoted versions are watched through burn-rate alerts. Escaped regressions and false alarms from every stage update the workload model, the baselines, the budgets and the alarm thresholds (Adegbite et al., 2025).

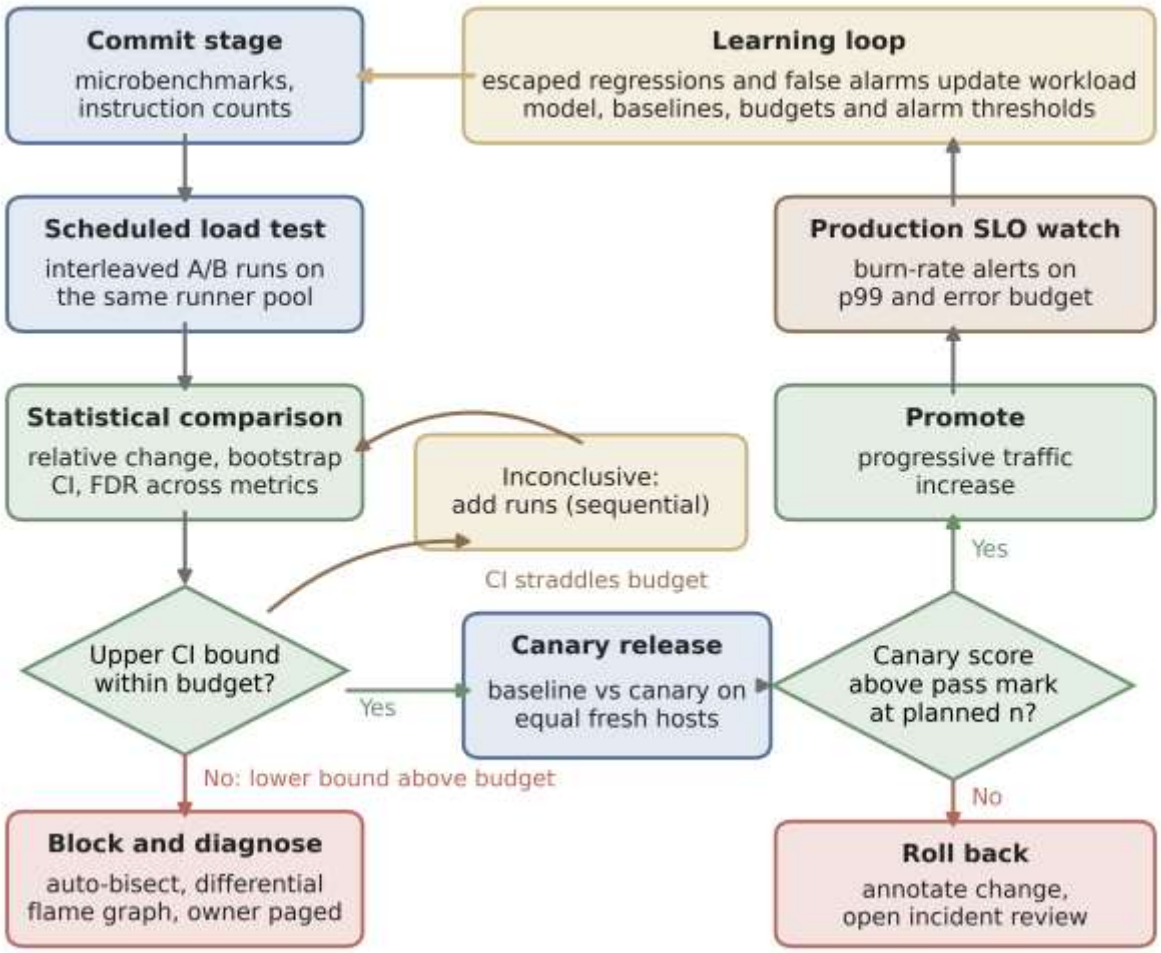


Figure 4. Gate decision flow across pipeline stages, with a three-way statistical rule, canary evaluation and a learning loop

The three-way rule is the element most often missing from practice. A binary gate forced to decide on an inconclusive interval either blocks on noise or passes on noise, and both outcomes erode trust. Making "inconclusive" an explicit state with a defined response, namely collecting more evidence, is the pipeline counterpart of accelerated-approval decision models that separate insufficient evidence from negative evidence (Eze et al., 2024b).

5.4 Propositions and Tests

The framework yields six propositions. Each names the evidence that would refute it.

P1 (noise dominance). For a given budget δ, both the false-alarm rate and the escaped-regression rate of a gate are governed by the ratio δ/σ_{eff}, where σ_{eff} is the standard error of the gate's comparison statistic measured through A/A runs; services whose ratio falls below 3 will show higher override rates and higher escape rates than services above it, whatever statistical test they use. Test: instrument A/A and injected-slowdown runs across services and regress override and escape rates on δ/σ_{eff}, controlling for change volume and service size. P1 is refuted if the choice of test explains more variance in escape rates than the noise ratio does (Komi, 2023).

P2 (pairing). At equal run counts, interleaved paired measurement on the same runner yields a lower A/A false-alarm rate than comparison with a stored baseline, because it removes the between-instance variance component. Test: within each service, alternate the two designs across builds at random and compare the proportion of A/A comparisons that fail. P2 is refuted if the difference is negligible on runner pools where the between-instance variance share, estimated by a random-effects model, is above 30 percent (Atta et al., 2022).

P3 (history over pairs). Change-point detection on build histories detects cumulative regressions composed of individually sub-budget steps that pairwise gates miss, at a cost of a bounded detection delay. Test: inject sequences of 0.5 to 1 percent slowdowns into replayed build histories and compare detection rates and delays for pairwise gates, CUSUM and E-Divisive detection. P3 is refuted if pairwise gates with anchored baselines match the history-based detectors on cumulative regressions (Atakpa et al., 2024).

P4 (fidelity before repetition). Escaped regressions concentrate in code paths and data regimes with low workload coverage, and once d' exceeds about 3, raising workload fidelity reduces escapes more than raising repetitions does. Test: map escaped regressions to profiling-based coverage at the time of release, then compare a staged workload refresh against an equal-cost increase in repetitions. P4 is refuted if escapes are unrelated to coverage (Dagodzo et al., 2020).

P5 (diagnosis and trust). Alarms delivered with attribution evidence (a commit range and a differential profile) are acted on faster and overridden less often than bare alarms of equal accuracy. Test: randomise the presentation of alarms at the alarm level and measure time to first action, override rate and the share of alarms closed as "could not reproduce". P5 is refuted if presentation has no effect once accuracy is held constant (Akintola et al., 2025; Ekelemu, 2025a).

P6 (anchored governance). Budgets anchored to a fixed reference release, with named owners, logged overrides and review dates, prevent the cumulative drift that rolling baselines allow. Test: an interrupted time-series comparison of long-run p99 trajectories for services that switch from rolling to anchored baselines against services that do not. P6 is refuted if anchored services show the same drift slope after the switch (Adebayo et al., 2025).

5.5 Rival Explanations and Boundary Conditions

Several simpler explanations compete with the proposed mechanism. Mature teams may adopt statistical gates and also write more efficient code, so an association between gating and fewer escaped regressions could reflect maturity rather than the gate. Lower change volume produces fewer regressions regardless of detection. Infrastructure upgrades that reduce noise can coincide with gate adoption. Announcing a gate may itself prompt more profiling and more careful review, an attention effect that fades. Reverse causation is also plausible: teams that have suffered performance incidents adopt gates because they are struggling, which would make gating look harmful in cross-sectional data. The validation design addresses these rivals through staged adoption, with each service serving as its own control, recorded concurrent changes, and outcome trends measured before adoption; the mechanism is supported only if intermediate indicators such as A/A false-alarm rates and detection delay change before downstream outcomes do (Amayo et al., 2023c).

Boundary conditions limit where the framework should be expected to work. Services whose latency is dominated by external dependencies cannot attribute changes to their own code without dependency virtualisation. Event-driven systems whose user-visible latency spans queues and several services need end-to-end rather than per-service budgets. Data-dependent performance, such as query plans that change with table statistics, resists test workloads that cannot reproduce production data skew. Low-traffic services cannot power canaries and must rely more heavily on pre-production evidence. At the other extreme, embedded and edge systems running on fixed hardware have little environmental noise and an explicit fidelity ladder from simulation to hardware-in-the-loop, so their binding constraint is workload and hardware fidelity rather than runner noise, while machine-learning inference on accelerators adds nondeterminism from kernel selection and batching (Oyesiji et al., 2026).

6 Discussion

The synthesis supports a conditional conclusion. Performance gating in continuous delivery can detect regressions early and at acceptable cost, but only when the gate is engineered as a detector, with its discriminability, error rates and decision rights made explicit. This section considers the implications for theory and practice and then examines the strongest arguments against the framework. The discussion is kept within the evidence: the propositions in Section 5.4 remain untested as a set, and the practical recommendations are framed as what the mechanism implies rather than as demonstrated effects. Sustaining such practices also depends on the capability of the engineers who operate them, since interpreting confidence intervals, profiles and change-point output is a learned skill that professional development must build deliberately (Ilodigwe et al., 2024b).

6.1 Theoretical Implications

Three implications follow for theory. First, framing the gate in signal detection terms dissolves a false dichotomy in the literature between "statistical" and "threshold" gates: every gate has a criterion and an implied false-alarm rate, and the only difference is whether these are known. This framing also supplies a common metric, d', with which studies conducted on different systems and runners can be compared, which the current literature lacks. Second, queueing theory implies that the sensitivity of a gate cannot be judged independently of the utilisation at which it measures, so studies of detection accuracy should report utilisation alongside noise. Third, the sociotechnical account adds a mediator that purely technical studies omit: trust, calibrated by experience of false

alarms and diagnostic support, determines whether detection becomes action. The framework thereby contributes to the theory of non-functional requirements by specifying how a performance requirement becomes a verifiable, traceable and governed property of each change, extending traceability models from functional to quantitative, probabilistic requirements (Ilodigwe et al., 2025a).

6.2 Practical Implications

Several practical steps follow from the mechanism. Teams should begin by measuring their gate rather than tuning it: a week of scheduled A/A comparisons establishes the effective noise, and a handful of injected slowdowns establishes the detection rate. Budgets should be set no tighter than about three times the effective standard error, and where that is looser than the business needs, the remedy is noise control and pairing rather than a stricter limit. Comparisons should report effect sizes with confidence intervals, should adopt a three-way rule with an explicit inconclusive state, and should anchor budgets to a reference release. Alarms should arrive with a commit range and a differential profile. Every budget should have a named owner, a review date and an override log. Runner images, instance types and kernel settings should themselves be declared and versioned as code, so that the measurement environment is reproducible and any change to it is reviewed like a change to the product, which is the governance logic that infrastructure-as-code practice applies to enterprise cloud deployments (Mbonu et al., 2020b). Benchmark suites should be pruned and prioritised by the code they cover and the history of what they have caught, drawing on established test-prioritisation methods (Anichukwueze et al., 2022).

A worked scenario makes the trade-offs concrete. Suppose a payments service carries a p99 budget of 5 percent per change, and A/A runs on shared runners show a single-run coefficient of variation of 8 percent against stored baselines. The power calculation in Section 4.2 then requires about 60 runs per side, which is prohibitive for a two-minute load test. Moving to interleaved pairs on the same runner reduces the effective coefficient of variation to 3 percent in this scenario, cutting the requirement to about 9 runs per side; spread across three parallel runners, each executing interleaved pairs of a two-minute test, the stage takes roughly twelve minutes of wall time, with a sequential stopping rule ending most comparisons earlier. The cost of runner time, and of dedicated runners where needed, should be accounted for explicitly and compared with the expected cost of escaped regressions, in line with cost-control frameworks for technology programmes (Adesuyi et al., 2023).

Observability data used for diagnosis requires governance of its own. Traces, profiles and replayed workloads may contain personal data, access tokens or sensitive business information, so redaction at collection, retention limits and role-based access are prerequisites rather than extras. Data protection impact assessment models and access control research apply directly, and where machine-learning anomaly detectors are introduced, organisational readiness for emerging AI regulation becomes a design consideration (Aliliele et al., 2025b).

6.3 Counter-Arguments

The first counter-argument holds that production is the only real test and that pre-production performance gating is wasted effort. The framework accepts part of this claim, namely that pre-production workloads never reach full fidelity, but rejects the conclusion: canaries have their own blind spots and expose users, and regressions that are large and exercised by the workload are cheaper to stop before release. The second holds that statistical gates slow pipelines unacceptably. The worked scenario suggests that paired, sequential designs keep the added delay to minutes, and advisory commit-stage checks add almost none; where delay is still binding, the gate can run asynchronously against merged changes and use history-based detection, accepting later detection in exchange for speed (Adegbite et al., 2025).

The third counter-argument predicts that machine-learning anomaly detection will make hand-specified statistics obsolete. Learned detectors can model seasonality and multivariate structure that simple tests ignore, but they inherit the noise and ground-truth problems described here, depend on carefully curated training data, since histories polluted by unlabelled infrastructure incidents teach a model that noise is signal, are harder to explain to the engineer who must act, and can perform unevenly across services in the way that accurate forecasting models can be unfair across regions (Komi, 2023). The fourth holds that the framework is too heavy for small teams. The constructs scale down: a small team can run A/A checks, pair its comparisons and name a budget owner at little cost, in the spirit of lightweight adaptations of structured improvement methods for resource-constrained organisations (Eyetsemitan et al., 2023b; Adebayo et al., 2025).

7 Limitations and Future Research Agenda

This review has limitations that qualify its conclusions, and it points to a research agenda that could test and refine the framework. Both are specific to performance evidence in delivery pipelines: the difficulty of establishing regression ground truth, the underreporting of industrial practice, and the dependence of results on runtime, architecture and traffic. The subsections below set out the limitations, a proposed validation design with its threats to validity, and the questions that most need answering. The agenda is ordered by dependency: shared datasets and ground-truth conventions come first, because without them detectors,

designs and governance practices cannot be compared across studies, and the propositions of Section 5.4 cannot be tested on more than one organisation at a time.

7.1 Limitations of the Review

The evidence base is heterogeneous in terminology, measurement environments and outcome definitions, which is why the review adjudicates mechanisms rather than pooling estimates. Publication bias is likely to be pronounced: detectors that work are reported, while gates that were abandoned because of noise rarely are, and industrial practice is described mostly in experience reports whose methods are incompletely disclosed. Few studies report A/A false-alarm rates, pipeline time cost or production follow-up, so several links in the mechanism rest on analytical argument from measurement and queueing theory rather than on direct observation. Cross-domain works were used for mechanism plausibility only, and their inclusion does not imply that effects transfer quantitatively. Finally, the propositions are untested as a set, and the thresholds they contain, such as d' of about 3 and a 30 percent between-instance variance share, are proposed starting points for empirical calibration rather than established constants (Sanni et al., 2022).

7.2 Validation Design and Threats to Validity

The recommended validation is a stepped-wedge study in which services adopt the NAPG components in randomised order across several periods, so that each service contributes both control and treatment observations and secular trends can be separated from effects. Primary outcomes are the escaped-regression rate, false alarms per 100 gated builds, time from alarm to attributed cause and pipeline delay per change; ground truth is established by injected slowdowns of known size and by production incidents confirmed through post-incident review. Controlled comparative trials in other applied sciences, which allocate units to alternative formulations and measure several outcomes jointly, illustrate the discipline required, and time-series methods for trend analysis supply the analytical tools (Atakpa et al., 2024). Table 4 lists the threats to validity specific to such a study and the corresponding mitigations.

Table 4. Threats to validity in a stepped-wedge evaluation of performance gating and their mitigations

Threat	How it arises in performance gating studies	Mitigation
Circular ground truth	Regressions are defined by the output of the detector under evaluation	Inject slowdowns of known size; confirm production regressions through post-incident review
Runner drift	The cloud provider changes processor generation or microcode during the study	Record processor model and microcode for every run; include as a covariate
Concurrent change	Infrastructure upgrades or traffic shifts coincide with gate adoption	Randomise adoption order; log concurrent events; track negative-control metrics
Attention effect	Engineers optimise because gates were announced, not because they block	Measure performance-work effort; compare with the pre-announcement trend
Benchmark survivorship	Noisy benchmarks are deleted, inflating apparent stability	Keep deleted benchmarks in the analysis with reason codes
Multiplicity and forking paths	Metrics, budgets and thresholds tuned after seeing results	Preregister primary metrics, budgets and false discovery rate level
Limited transfer	Results from one runtime, architecture or traffic level generalised to others	Stratify by runtime (managed or native), architecture and traffic volume
Telemetry exposure	Replayed workloads, traces and profiles contain personal or sensitive data	Redact at collection; limit retention; restrict access by role

The threats in Table 4 are not interchangeable. A very large build history does not repair circular ground truth, and a strong detection rate does not compensate for silent overrides or for exposure of personal data in traces. Each threat therefore needs its own mitigation and its own reporting (Fadayomi et al., 2019).

7.3 Future Research Agenda

Five research directions follow. First, the field needs open, well-documented datasets of repeated performance measurements from real pipelines, including A/A runs, injected regressions and runner metadata, so that detectors can be compared on common

ground. Second, detectors should be developed and evaluated for the skewed, heavy-tailed and bimodal distributions that cloud benchmarks actually produce, with particular attention to tail statistics. Third, adaptive allocation of repetitions, in which sequential tests stop early when evidence is clear and invest runs where it is not, could reduce pipeline cost substantially and deserves formal cost-benefit analysis. Fourth, energy and carbon per request are emerging as performance budgets in their own right, and the framework's constructs extend naturally to them, as maturity models for carbon management in engineering delivery suggest. Fifth, the human factors of gate trust, including how engineers interpret confidence intervals, when they override and how ownership is negotiated, remain almost unstudied, even though the mechanism depends on them (Abdulhadi et al., 2026b; Obogo et al., 2026; Aminu-Ibrahim et al., 2025c).

8 Conclusion

Performance regressions escape continuous delivery pipelines not because tools are lacking but because performance gates are usually built as static limits on noisy measurements of unrepresentative workloads, with no stated error rates and no clear owner. This review has argued that a gate should be treated as a detector. Its discriminability is set by workload fidelity and noise control, its decisions by a statistical rule that reports effect sizes, controls multiplicity, admits an inconclusive outcome and watches build histories for creeping change, and its value by the speed with which alarms are diagnosed and the authority with which they are acted upon. The Noise-Aware Performance Gating framework assembles these elements into a mechanism with measurable indicators, six falsifiable propositions and a stepped-wedge validation design. Teams that measure their gates with A/A comparisons and injected slowdowns, pair their measurements, test tail percentiles against anchored budgets and assign every budget an owner will know how often their gate is wrong, which is the precondition for trusting it when it is right.

Funding: This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Competing interests: None declared.

Data availability: No new data were generated or analysed in this study; all sources are cited. Figure 2 is an illustrative simulation whose parameters are stated in the text.

References

Abdulhadi, B. M., & Hamdan, A. M. (2026b). Net-zero project delivery: A maturity model for carbon management capability in construction organizations. International Journal of Engineering and Modern Technology, 12(6), 199-233. https://doi.org/10.56201/ijemt.vol.12.no6.2026.pg199.233

Adebayo, A., Amadi, C., & Ijagbemi, A. O. (2025). From audit to continuous control: Implementation pathways, governance, and organizational readiness for AI-enabled compliance in critical infrastructure. International Journal of Computer Science and Mathematical Theory, 11(8), 88-134. https://doi.org/10.56201/ijcsmt.vol.11.no8.2025.pg88.134

Adebayo, A., Amadi, C., & Ijagbemi, A. O. (2026). Semantic de-duplication of shared cyber threat indicators using large language model embeddings. Global Journal of Engineering and Technology Review, 2(9), 465-482. https://doi.org/10.65150/EP-gjetr/V2E9/2026-07

Adegbite, M. P., Adebayo, A., & Ahmed, M. O. (2025). A cyber risk quantification and governance architecture for critical infrastructure: From posture measurement to executive reporting. International Journal of Computer Science and Mathematical Theory, 11(12), 232-293. https://doi.org/10.56201/ijcsmt.vol.11.no12.2025.pg232.293

Adelanwa, A., Basnet, A., & Anene, U. N. (2023a). Data driven digital transformation models for lifecycle performance management in infrastructure delivery. International Journal of Advanced Multidisciplinary Research and Studies, 3(6), 2646-2662. https://doi.org/10.62225/2583049X.2023.3.6.5968

Adesuyi, M. O., Kalu, A., & Walawalkar, G. (2023). Data-led cost governance in technology-intensive enterprises. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 9(3), 877-896. https://doi.org/10.32628/IJSRCSEIT

Akintola, A. S., Fawehinmi, Y. O., Aiyenitaju, O., Chinemerem, B., & Emmanuella, O. (2025). Digital transformation in telehealth: A systematic review of user trust, privacy, and regulatory governance in AI-powered remote monitoring systems. Journal of Scientific Research and Reports, 31(11), 163-182. https://doi.org/10.9734/jsrr/2025/v31i113658

Aliliele, C., Mbonu, I. S., & Iwuanyanwu, U. (2024a). A conceptual framework for enterprise data sensitivity classification and regulatory traceability mechanisms. International Journal of Advanced Multidisciplinary Research and Studies, 4(6), 3103-3124. https://doi.org/10.62225/2583049X.2024.4.6.5991

Aliliele, C., Mbonu, I. S., Uzoka, E., & Iwuanyanwu, U. (2025b). Advances in data lakehouse governance architectures for enterprise data loss prevention and compliance assurance. Shodhshauryam, International Scientific Refereed Research Journal, 8(4), 171-213. https://doi.org/10.32628/SHISRRJ258474

Amarahobu, C., Ekelemu, O., Akano, O. A., & Adikwu, F. E. (2025a). Structured mentorship models in on-the-job technician development: A review of advancement and promotion coaching schemes, workplace learning theory and the leadership interface in continuous processing environments. International Journal of Social Sciences and Management Research, 11(8), 626-656. https://doi.org/10.56201/ijssmr.vol.11no8.2025.pg626.656

Amayo, E. B., Owulade, O. A., & Isi, L. R. (2023c). The role of diversity and inclusion in enhancing project team performance and delivery in multinational environments. Journal of Frontiers in Multidisciplinary Research, 4(1), 48-58. https://doi.org/10.54660/.IJFMR.2023.4.1.48-58

Aminu-Ibrahim, A. Y., Ogbete, J. C., & Ambali, K. B. (2018). Developing sustainable diagnostic laboratory infrastructure models for emerging and resource constrained health systems. Iconic Research and Engineering Journals, 1(8), 118-132. https://doi.org/10.64388/IREV1I8-1713586

Aminu-Ibrahim, A. Y., Ogbete, J. C., & Ambali, K. B. (2024). Governance and accountability models for public private partnerships in healthcare infrastructure development. International Journal of Advanced Multidisciplinary Research and Studies, 4(6), 2943-2960. https://doi.org/10.62225/2583049X.2024.4.6.5699

Aminu-Ibrahim, A. Y., Ogbete, J. C., & Iwuanyanwu, O. C. (2025c). Sustainable healthcare infrastructure performance metrics for long-term asset management and value creation. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 11(4), 566-601. https://doi.org/10.32628/CSEIT251116277

Anichukwueze, C. C., Ominyi, M., Uzougbo, N. S., & Jones, B. C. (2022). Cross-functional legal education as risk mitigation: Empirical analysis of contract error reduction in engineering teams. International Journal of Engineering and Modern Technology, 8(5), 142-180. https://doi.org/10.56201/ijemt.v8.no5.2022.pg142.180

Arumosoye, O. M., & Obriki, O. D. (2021). Organizational learning-based conceptual maturity model for continuous safety performance improvement. International Journal of Multidisciplinary Research and Growth Evaluation, 2(1), 970-980. https://doi.org/10.54660/.IJMRGE.2021.2.1.970-980

Atakpa, M. I., Abolaji, T. O., & Abetoh, N. F. (2024). Statistical time-series models for long-range public health trend forecasting: A review and conceptual framework. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 10(6), 2748-2780. https://doi.org/10.32628/CSEIT2410792

Atta, S., Adjaklo, E., Asomani, E., & Adenuga, O. M. (2022). Advances in welding inspection standards and quality assurance practices in large-scale power generation projects. International Journal of Engineering and Modern Technology, 8(5), 101-124. https://doi.org/10.56201/ijemt.v8.no5.2022.pg101.124

Dagodzo, D., & Ahiaeke Patrick, M. C. (2020). UAV-based pipeline and corridor monitoring: A review of current practices and emerging technologies. Iconic Research and Engineering Journals, 3(10), 574-597. https://doi.org/10.64388/IREV3I10-1716084

Ekelemu, O. (2025a). Control room alarm management and human factors in large liquefied natural gas facilities: A comprehensive review of rationalization frameworks, operator response, and the elimination of standing and inhibited alarms. International Journal of Engineering and Modern Technology, 11(8), 352-396. https://doi.org/10.56201/ijemt.vol.11.no8.2025.pg352.396

Eyetsemitan, R. A., Ambali, K. B., Oyeleye, A. O., & Fadayomi, O. (2023b). User acceptance testing in small business technology deployment: A structured validation framework for lean operational environments. International Journal of Multidisciplinary Research and Growth Evaluation, 4(6), 1512-1531. https://doi.org/10.54660/IJMRGE.2023.4.6.1512-1531

Eze, F. I., Akinleye, O. K., & Anene, U. N. (2024b). Development of a risk-based accelerated approval decision model balancing speed, safety, and evidence quality in pharmaceutical regulatory science. Research Journal of Pure Science and Technology, 7(6), 120-161. https://doi.org/10.56201/rjpst.v7.no6.2024.pg120.161

Eze, F. I., Anene, U. N., & Akinleye, O. K. (2022b). Design of an integrated regulatory readiness maturity model for pharmaceutical companies seeking global market entry. International Journal of Health and Pharmaceutical Research, 7(1), 77-112. https://doi.org/10.56201/ijhpr.v7.no1.2022.pg77.112

Fadayomi, O., Abolaji, T. O., Edivri, J., Ogbole, J. I., Okoruwa, P. O., & Akeju, B. (2019). Risk-based cybersecurity assurance and data availability: Limitations, advances and future research opportunities. Iconic Research and Engineering Journals, 2(12), 602-617. https://doi.org/10.64388/IREV2I12-1713779

Gideon, E. N., Adaramola, T. S., & Fadero, S. (2019). Reliability engineering and failure analysis in complex engineering systems. Iconic Research and Engineering Journals, 2(11), 703-727. https://doi.org/10.64388/IREV2I11-1722490

Ilodigwe, L., & Adesemoye, A. C. (2024a). Advances in real-world evidence, predictive analytics, and data-driven decision making for improving health system performance and patient outcomes. International Journal of Health and Pharmaceutical Research, 9(5), 174-197. https://doi.org/10.56201/ijhpr.v9.no5.2024.pg174.197

Ilodigwe, L., & Adesemoye, A. C. (2025a). Advances, risks, and implementation challenges of artificial intelligence as a force multiplier for clinical decision making and health system efficiency. International Journal of Medical Evaluation and Physical Report, 9(7), 165-185. https://doi.org/10.56201/ijmepr.v9.no7.2025.pg165.185

Ilodigwe, L., & Adesemoye, A. C. (2024b). Beyond technology: A strategic framework for building sustainable and resilient health systems through organizational strategy, operational excellence, and workforce capability. International Journal of Medical Evaluation and Physical Report, 8(6), 299-319. https://doi.org/10.56201/ijmepr.v8.no6.2024.pg299.319

Komi, N. M., & Adamolekun, A. (2021). Interpretable machine learning for early failure prediction in distributed renewable energy assets. International Journal of Multidisciplinary Research and Growth Evaluation, 2(6), 1015-1038. https://doi.org/10.54660/.IJMRGE.2021.2.6.1015-1038

Komi, N. M. (2023). When accurate forecasts are not fair forecasts: Deep learning load prediction and equity across districts. Shodhshauryam, International Scientific Refereed Research Journal, 6(6), 476-546. https://doi.org/10.32628/SHISRRJ247102

Mbonu, I. S., Iwuanyanwu, U., Aliliele, C., & Uzoka, E. (2020b). Advances in infrastructure as code governance for secure terraform based enterprise cloud deployments. International Journal of Multidisciplinary Research and Growth Evaluation, 1(5), 811-828. https://doi.org/10.54660/.IJMRGE.2020.1.5.811-828

Obogo, S. F., Nwafor, M. I., & Ozobu, C. O. (2023). Conceptual leadership model for safety culture development in construction and engineering projects. International Journal of Scientific Research in Civil Engineering, 7(6), 121-153. https://doi.org/10.32628/IJSRCE237554

Obogo, S. F., Ozobu, C. O., Garba, B. M. P., & Adio, S. A. (2026). Predictive safety analytics model for early detection of high-risk construction activities. Global Journal of Engineering and Technology Review, 2(3), 92-109. https://doi.org/10.65150/EP-gjetr/V2E3/2026-03

Ogbole, J. I., Okoruwa, P. O., Babatope, O. M., & Oyewole, T. (2021). Developing an integrated data visualization model for continuous business performance monitoring and optimization. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 7(5), 441-467. https://doi.org/10.32628/IJSRCSEIT

Ogundairo, K. M., & Ayivi-Donkor, S. S. (2023b). Telemetry-driven value realization: Infusing economic evaluation algorithms directly into enterprise data pipelines. International Journal of Economics and Financial Management, 8(8), 195-232. https://doi.org/10.56201/ijefm.v8.no8.2023.pg195.232

Olagunju, E. O., Edivri, J., & Oteri, O. (2024). Applied performance optimization frameworks for managing high-demand enterprise technology programs and system rollouts. International Journal of Advanced Multidisciplinary Research and Studies, 4(6), 3059-3070.

Oluwo, K., Bola-Sadipe, D., & Aliliele, C. (2025). Conceptual framework for data analytics driven financial decision making in banking sector organizations. Shodhshauryam, International Scientific Refereed Research Journal, 8(5), 259-302. https://doi.org/10.32628/SHISRRJ258529

Oyesiji, S. O., Anunagba, C. O., Nwakamma, S., & Sankara, H. W. Y. C. (2026). Automated validation in embedded IoT development: A review of continuous integration and delivery across the fidelity ladder. Journal of Engineering Research and Reports, 28(9), 145-157. https://doi.org/10.9734/jerr/2026/v28i92005

Sakyi, J. K., Filani, O. M., Nnabueze, S. B., Okojie, J. S., & Ogedengbe, A. O. (2022b). Developing KPI frameworks to enhance accountability and performance across large-scale commercial organizations. Journal of Frontiers in Multidisciplinary Research, 3(2), 81-93. https://doi.org/10.54660/.IJFMR.2022.3.2.81-93

Sanni, J. O., Atima, M. E., & Attah, A. (2022). Systematic review of attribution modeling methods resolving bias in multi-touch journeys. Shodhshauryam, International Scientific Refereed Research Journal, 5(1), 304-334. https://doi.org/10.32628/SHISRRJ247135