

Evidence-Aware Predictive Delivery and Continuous Software Assurance for Regulated Energy Utility Platforms

Suresh Kumar Raman¹, Rajkamal Selvaraj Shauryan²

¹Department of CSE, Jain Deemed-to-be University, Bengaluru, India

²Department of CSE, Jain Deemed-to-be University, Bengaluru, India

sureshr0287@gmail.com, rajshauryan@gmail.com

Abstract - Energy utility platforms increasingly depend on software-intensive systems for grid analytics, outage management, market operations, distributed-energy-resource coordination, customer services, and telemetry dashboards. CI/CD can improve deployment velocity, but release governance in regulated utility environments often remains static. This study develops an evidence-aware release-assurance framework combining multi-signal risk estimation, machine-verifiable evidence, rollback readiness, deterministic policy routing, and utility operating context. The evaluation uses real ISO New England observations and transparent synthetic CI/CD and release telemetry because public utility datasets do not contain proprietary software-delivery records. Four configurations are compared: static governance, a policy-only gate, a prior predictive-delivery benchmark, and the proposed utility-aware framework. Learned risk estimation improves discrimination, while the proposed framework adds evidence, rollback, policy, and operating-context constraints. The results support treating regulated utility release assurance as a combined prediction, evidence, recovery, and governance problem.

Key Words: energy utilities, release-risk prediction, software assurance, CI/CD, operational technology, evidence-aware governance.

I. INTRODUCTION

Energy utilities increasingly operate through software. Grid-operations analytics, outage-management applications, distributed-energy-resource systems, market interfaces, customer platforms, cloud monitoring services, and operational dashboards all depend on frequent software and configuration changes. These changes support modernization but create a release-governance problem. A software change may pass automated tests and still be unsuitable for fast deployment if rollback is weak, evidence is incomplete, policy status is unclear, or the change is scheduled during a sensitive operating period.

The energy operating context changes over time. The U.S. Energy Information Administration (EIA) Form EIA-930 provides hourly balancing-authority operating data, including demand, forecast demand, net generation, and interchange information [1]. This study uses an ISO New England (ISO-NE) operating extract derived from that public grid-monitoring data as the utility-context input. Operating context is not treated as a direct grid-reliability label. It is used to determine whether a software release occurs during normal or elevated operating conditions.

Regulated utility environments also require disciplined configuration and supply-chain controls. North American Electric Reliability Corporation (NERC) CIP-010-4 addresses configuration-change management and vulnerability-

assessment expectations for applicable Bulk Electric System (BES) cyber systems [2]. NERC CIP-013-2 addresses cybersecurity supply-chain risk management [3]. These controls do not prescribe a predictive release model, but they reinforce why configuration integrity, traceability, tested change, and supply-chain evidence matter in utility software delivery.

Conventional release governance often relies on static classification. A change may be labeled standard, normal, emergency, low risk, or high risk, with approval requirements assigned by category. Static classifications are auditable, but they do not always respond to the actual risk state of a release candidate. Two releases in the same category may differ substantially in code churn, test confidence, recent CI instability, runtime dependency health, evidence completeness, rollback readiness, and operating context.

This paper is a reproducible simulation and architecture study, not a production validation study. The utility operating context is real, but the software-release records, CI/CD variables, rollback states, evidence states, and outcome labels are generated because public utility datasets do not contain proprietary delivery telemetry. The study therefore evaluates a utility-specific architecture, signal structure, policy-routing logic, and reproducibility method that can later be validated in shadow mode using de-identified real utility CI/CD, deployment, rollback, and incident records.

Prior research addresses change-level risk [4], [5], CI behavior [6], [8], secure development and provenance [7], [10], platform evidence [11], OT security [9], utility situational awareness [12], and risk-tiered governance [15], but largely as separate concerns. The gap is a utility-specific release-assurance design that combines these concerns without using grid state as a software-failure label. This study contributes a utility operating-sensitivity representation from real ISO-NE observations, an assurance score joining risk and control state, deterministic utility-aware routing, and a reproducible evaluation using comparative modeling, ablation, evidence degradation, and shifted-risk testing.

II. LITERATURE REVIEW

A. Change-Risk and CI/CD Signals

Change-risk modeling is relevant because release risk is often concentrated in particular code, dependency, and process patterns. Kamei et al. showed that just-in-time quality assurance can predict defect-inducing changes using change and developer/process signals [4]. Rausch et al. identified recurring CI build-failure patterns [8], while TravisTorrent links build metadata, commits, logs, and project context for large-scale CI research [13]. These studies support using observable change and pipeline behavior rather than only static release categories.

Predictive delivery research motivates combining multiple signal families. Prior work organizes release-risk information across change, pipeline, runtime, and governance dimensions [5]. Huang et al. show that some legitimate CI failures are unrelated to the triggering patch [6], cautioning against single-signal defect labels. These findings inform the present multi-signal model, while the method in [5] is retained as one external benchmark.

B. Software Assurance, OT, and Utility Context

Software assurance in regulated environments requires more than test success. Evidence must show what changed, which policies were evaluated, which artifact was deployed, who owned the decision, what rollback path existed, and whether controls operated as expected. The NIST Secure Software Development Framework (SSDF) recommends integrating secure development practices into the software lifecycle [7]. Rausch et al. showed that CI build failures follow identifiable patterns, which supports structured analysis of pipeline evidence rather than ad hoc release decisions [8].

Utility software operates near operational technology (OT) environments and operational processes. NIST SP 800-82 Rev. 3 frames OT security around mission, reliability, and safety constraints [9]. Supply-chain Levels for Software Artifacts (SLSA) provides provenance expectations for software artifacts [10]. These sources support machine-verifiable delivery evidence and the use of utility operating context as a routing signal while preserving a clear boundary: grid data affects release tolerance but does not label software-release success or failure.

C. Platform Evidence and Risk-Tiered Governance

Machine-verifiable platform evidence provides a second foundation for release assurance. NIST SSDF and SLSA establish secure-development and provenance expectations [7], [10]. Platform-engineering research shows how standardized delivery paths, policy-as-code, and evidence-by-construction can generate auditable artifacts [11]. The present study converts these evidence concepts into a weighted completeness score using test, policy, provenance, deployment, approval, ownership, exception, and rollback artifacts.

Utility release decisions also need operational context rather than software evidence alone. The National Cybersecurity Center of Excellence (NCCoE) describes electric-utility situational awareness through operational data collection and analysis [12]. The ISONE Smart City Energy Dataset provides a secondary utility profile for robustness and leakage checks [14]. These sources support using demand, ramp, forecast error, generation, interchange, and volatility as operating-state inputs, while preserving a strict boundary: utility observations can alter release tolerance but do not serve as software-failure labels.

Release governance should match risk and validation need. Prior regulated CI/CD research identifies risk-tiered governance, execution validation, evidence-by-design, and accountable exceptions as complementary controls [15]. The present study defines four deterministic paths: fast, guarded, hold, and exception, using explicit thresholds on predicted risk, evidence completeness, rollback readiness, policy state, and utility context. The learned model estimates risk; the policy gate retains authority over the final route.

D. Energy Utility Operating Data

The primary energy context comes from ISO-NE observations derived from EIA-930 public grid-monitoring data [1]. A

secondary ISONE Smart City Energy Dataset is used for robustness profiling and leakage checks [14]. DOE OE-417 records are identified as a future event-window validation source [16]. Public utility datasets do not contain Git commits, CI/CD runs, release approvals, rollback records, or software incident labels, so the study separates real energy operating context from synthetic software-release telemetry and does not treat grid data as a software-failure label.

III. METHODOLOGY

A. System Overview

The proposed framework contains six layers: a standardized delivery layer, a release-risk signal layer, a utility operating-context layer, an evidence and rollback assurance layer, a predictive assurance and policy-gate layer, and a deployment validation and feedback layer. A release enters through a controlled delivery path that produces test outputs, policy decisions, provenance data, deployment metadata, ownership information, and rollback artifacts. The release-risk model estimates failure probability from software-delivery signals. The assurance layer combines that estimate with evidence completeness, rollback readiness, policy status, and utility operating context. A deterministic policy gate then assigns the release to a governance path, and post-deployment health and incident observations provide feedback for later threshold review.

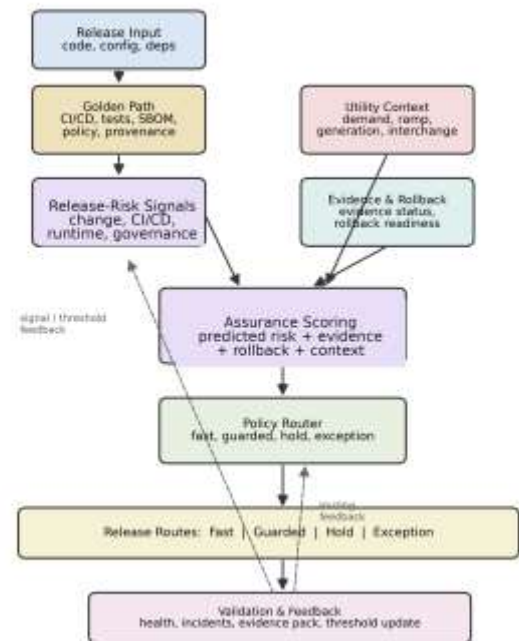


Fig. 1. Evidence-aware predictive delivery architecture. Release inputs, generated evidence, multi-signal release-risk information, utility context, and rollback readiness feed assurance scoring and deterministic policy routing.

B. Multi-Signal Release-Risk Model

For release candidate j , the release-risk feature vector is defined in Eq. (1), where X_{chg} represents change and code signals, X_{ci} represents CI/CD signals, X_{run} represents runtime and Site Reliability Engineering (SRE) signals, and X_{gov} represents governance and evidence signals.

$$X_j^R = [X_j^{chg}, X_j^{ci}, X_j^{run}, X_j^{gov}] \quad (1)$$

Representative change features include normalized churn, critical-file modification, dependency delta, configuration-file

change, and ownership dispersion. CI/CD features include recent build failure rate, retry frequency, test instability, pipeline-duration drift, and environment mismatch. Runtime features include recent service-level objective (SLO) degradation, dependency health, incident adjacency, error-budget pressure, and previous rollback history. Governance features include policy-check status, evidence availability, approval state, segregation-of-duties signal, and exception-state information.

$$p_{\text{hat}_j} = P(Y_j = 1 | X_j^R) \quad (2)$$

$$\text{logit}(p_{\text{hat}_j}) = \beta_0 + \beta_c X_j^{\text{chg}} + \beta_i X_j^{\text{ci}} + \beta_r X_j^{\text{run}} + \beta_g X_j^{\text{gov}} \quad (3)$$

Equations (1)-(3) define the learned release-risk component and permit signal-family ablation. Signal selection is informed by change-risk and CI research [4], [6], [8], [13] and predictive-delivery work [5]. The method in [5] is an external benchmark; the proposed model combines its learned estimate with utility-context and assurance variables defined in Sections III-C and III-D. L2-regularized logistic regression permits inspection of fitted weights, calibration, and feature-family removal.

C. Utility Context and Assurance Scoring

For each release timestamp t , the utility-context vector is defined in Eq. (4). Demand, forecast demand, forecast error, ramp, generation, interchange, and rolling volatility are treated as operating-context inputs rather than labels of release success or failure.

$$U_t = [D_t, F_t, \epsilon_t, R_t, G_t, I_t, V_t] \quad (4)$$

$$C_t = 0.20z_D + 0.20z_{\epsilon} + 0.20z_R + 0.15z_G + 0.15z_I + 0.10z_V \quad (5)$$

The utility operating-sensitivity score C_t does not measure grid reliability directly. It indicates whether a software release occurs during a more sensitive operating context. The same release may therefore receive different routing treatment under normal conditions than under high demand, elevated forecast error, rapid load ramp, or greater short-term volatility.

Evidence completeness is computed from delivery artifacts, as shown in Eq. (6). Items include a test report, policy evaluation, deployment manifest, software bill of materials (SBOM), artifact provenance, accountable owner, approval record, exception record where applicable, and rollback validation. The score measures whether decision evidence is present rather than whether the release is predicted to succeed.

$$E_j = \text{sum}(w_m a_{jm}) / \text{sum}(w_m) \quad (6)$$

$$A_j = 0.35(1 - p_{\text{hat}_j}) + 0.25E_j + 0.15B_j + 0.15P_j + 0.10(1 - C_t) \quad (7)$$

Equation (7) fixes the assurance-score weights before final testing. Low predicted failure probability does not establish release assurance when evidence, rollback, policy, or operating context is weak; the learned estimate cannot override mandatory controls.

D. Dynamic Governance Paths

The deterministic gate routes releases into four paths defined for this study. Fast Path is used when $p_{\text{hat}} \leq 0.30$, $A \geq 0.75$, $E \geq 0.85$, $B \geq 0.80$, $P = 1$, and $C \leq 0.70$. Guarded Path is used for moderate risk when mandatory policy conditions pass but added validation or progressive rollout is required. Hold Path is used when $p_{\text{hat}} > 0.55$, $A < 0.60$, $E < 0.65$, $B < 0.55$, or $P = 0$.

Exception Path requires an accountable owner, rationale, expiration, and post-execution evidence. The learned model can influence risk scoring, but it cannot bypass mandatory policy controls.

E. Datasets and Validity Controls

The primary utility-context dataset is an ISO-NE operating extract with 4,417 hourly observations from 07-01-2021 through 01-01-2022. The secondary robustness dataset is the ISONE Smart City Energy Dataset, with 72,960 half-hour records from 01-01-2021 through 02-28-2025 and 60 variables [14]. The software-delivery dataset contains 12,000 reproducible synthetic release events because public utility datasets do not contain source-code changes, CI/CD executions, release approvals, rollback records, or software incident labels. DOE OE-417 records are identified as an event-validation source for future outage or disturbance windows [16].

TABLE I. DATASET USE AND MODELING ROLE

Source	Use	Validity boundary
EIA-930 / ISO-NE	4,417 hourly grid observations	No release telemetry
ISONE Smart City [14]	72,960 half-hour records	Derived fields require checks
Synthetic releases	12,000 release events	Simulated labels
DOE OE-417 [16]	Future event-window source	Not release labels

Table I separates real energy operating data from synthetic delivery records. The secondary ISONE dataset was inspected before modeling: Net Load equals Electricity Load minus Renewable Energy Load; Peak Load Indicator equals the highest five percent of historical load; and Curtailment Risk or Surplus Flag equals one when Net Load is negative. These deterministically derived variables are not used as supervised targets while retaining their determining inputs, avoiding target leakage and preventing circular validation.

TABLE II. FIXED SIMULATOR AND ROUTING PARAMETERS

Element	Fixed setting
Seed and split	Seed 20260817; 70/15/15 train/validation/test; 500 bootstrap samples
Model	L2 logistic regression; $C=1.0$; max_iter=1000; train-set scaling
Latent label	$\eta = -2.25 + 0.90\text{chg} + 0.55\text{ci} + 0.70\text{run} + 0.30(1-E) + 0.25\text{app} + 0.35C + u$
Noise	$u \sim N(0, 0.35)$; outcome sampled from Bernoulli(sigmoid(η))
Routing	Fast and hold thresholds fixed before final testing

Table II reports the fixed seed, label-generation process, model settings, and routing thresholds. These controls are reported because software-release outcomes are generated. The design evaluates architectural behavior, signal contribution, routing tradeoffs, and reproducibility under controlled conditions; it does not establish production generalization.

F. Configurations and Tests

Four configurations are evaluated. Static Governance uses fixed classification and approvals. A Policy-Only Assurance Gate applies deterministic evidence, rollback, policy, and utility-context rules without a learned model. A prior predictive-delivery benchmark implements the signal families in [5]. The Proposed Framework combines its multi-signal risk estimate

with evidence completeness, rollback readiness, policy routing, and utility context. The benchmark is an external learned comparison, not the basis of the proposed routing architecture.

TABLE III. COMPARATIVE EXPERIMENTAL CONFIGURATIONS

Configuration	Inputs	Routing objective
Static	Manual class, criticality	Fixed approvals
Policy-only	Evidence, rollback, policy	Rule-based paths
PDI [5]	Change, CI/CD, runtime, governance	Learned benchmark
Proposed	Risk, evidence, rollback, policy, context	Evidence-aware paths

The study reports ROC-AUC, F1-score, false-negative rate, unsafe fast-path rate, unnecessary manual-review rate, policy escape rate, evidence completeness, and rollback-readiness coverage. Signal-family ablation removes change, CI/CD, runtime, evidence, and utility-context signals one family at a time. Evidence-degradation testing removes assurance artifacts, and a shifted-risk test evaluates stability when the test distribution differs from the original latent risk function.

IV. RESULTS

A. Prediction and Governance Results

TABLE IV. PRIMARY PREDICTION AND GOVERNANCE RESULTS

Model	AUC / F1	FN / Unsafe	Review / Escape
Static	0.726 / 0.532	44.2% / 28.8%	11.4% / 66.3%
Policy-only	0.704 / 0.499	42.4% / 4.9%	3.7% / 0.0%
PDI	0.805 / 0.600	37.5% / 13.9%	9.7% / 0.0%
Proposed	0.810 / 0.602	36.4% / 13.1%	17.0% / 0.0%

Table IV shows that learned configurations materially improve release-risk discrimination compared with static governance. The PDI benchmark raises ROC-AUC from 0.726 to 0.805 and F1-score from 0.532 to 0.600. The proposed framework reaches 0.810 and 0.602, respectively. Its main distinction is therefore not large predictive superiority, but the addition of evidence, rollback, policy, and utility-context constraints to routing.

The policy-only gate separates deterministic assurance from learned prediction. It produces a lower ROC-AUC than the learned configurations but prevents policy escape because mandatory evidence and policy failures are blocked by design. The proposed framework combines the two behaviors: learned signals improve discrimination, while deterministic controls prevent a low predicted risk score from overriding missing evidence, inadequate rollback readiness, failed policy state, or elevated operating sensitivity.

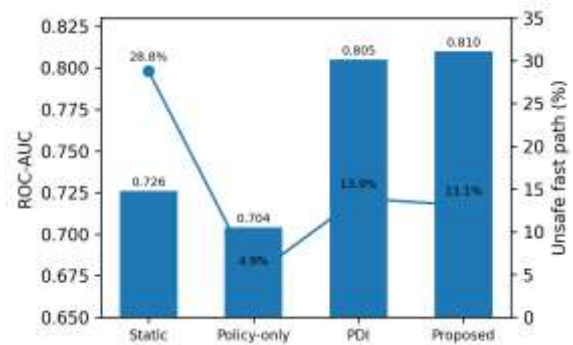


Fig. 2. Primary model comparison. The bars show ROC-AUC, while the line shows unsafe fast-path rate. Learned risk estimation improves discrimination; deterministic assurance controls constrain unsafe routing.

B. Ablation and Learned-vs.-Deterministic Effects

TABLE V. SIGNAL-FAMILY ABLATION RESULTS

Variant	AUC	F1	Escape
Full	0.810	0.602	0.0%
-Change	0.772	0.568	0.0%
-CI/CD	0.804	0.604	0.0%
-Runtime	0.788	0.593	0.0%
-Evidence	0.804	0.596	68.6%
-Utility	0.808	0.594	0.0%

Table V shows that change signals are the strongest predictive contributor; removing them reduces ROC-AUC from 0.810 to 0.772. Runtime signals also matter. Evidence removal causes only a modest predictive drop, but policy escapes rise to 68.6 percent. Evidence therefore functions primarily as a governance-control signal in this design, while change and runtime information contribute more directly to predictive discrimination.

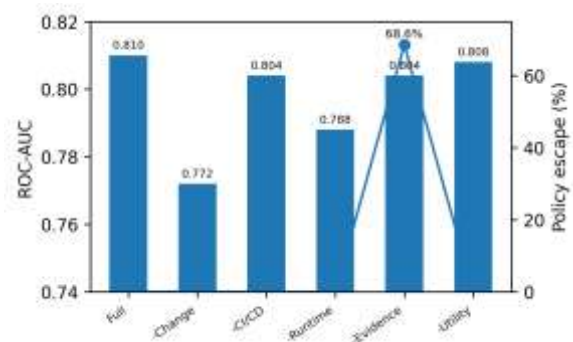


Fig. 3. Ablation impact on ROC-AUC and policy escape rate. Removing evidence has limited effect on AUC but creates substantial policy escape, separating predictive and deterministic effects.

The evaluation therefore distinguishes learned prediction effects from deterministic governance effects. ROC-AUC, F1-score, and false-negative rate depend on the fitted risk model. Policy escape, mandatory evidence blocking, and exception routing are substantially determined by the policy gate. The 0.0 percent policy-escape result is consequently evidence of deterministic control enforcement, not evidence that the statistical model learned to prevent policy violations.

The ablation results show why predictive quality and control assurance require separate metrics. Removing change or runtime information reduces discrimination, while removing evidence has only a modest AUC effect but creates substantial policy

escape. Learned risk estimation and deterministic assurance therefore serve complementary functions.

C. Utility-Context Sensitivity

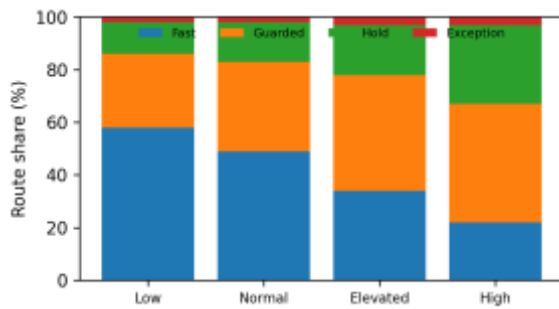


Fig. 4. Governance path distribution by utility operating-sensitivity tier. Fast-path routing decreases as operating sensitivity rises, while guarded and hold routing increase.

Figure 4 shows the utility-specific effect of operating context. Fast-path routing decreases as utility sensitivity rises, while guarded and hold routing increase. Low-risk, policy-compliant, well-evidenced releases can still use the fast path during normal conditions, but comparable software risk may receive more conservative treatment during sensitive periods. The decision rule also supports a controlled counterfactual interpretation: holding software-risk, evidence, rollback, and policy inputs constant, increasing C_t lowers the assurance score in Eq. (7) and can remove Fast Path eligibility above 0.70. This is a property of the routing rule, not a production validation result.

D. Evidence Stress and Robustness

TABLE VI. EVIDENCE-DEGRADATION STRESS TEST

Drop	Fast / Hold	Unsafe	Review / Latency
0%	41.6% / 26.8%	13.1%	17.0% / 22 min
10%	37.7% / 34.0%	11.7%	25.2% / 22 min
20%	32.7% / 41.9%	10.0%	34.8% / 22 min
30%	28.8% / 48.7%	8.6%	41.7% / 22 min
40%	24.1% / 56.9%	8.3%	51.6% / 95 min

Table VI removes selected assurance artifacts before routing. As evidence loss increases, fast-path eligibility falls from 41.6 percent to 24.1 percent, while hold routing rises from 26.8 percent to 56.9 percent. Unsafe fast-path routing declines, but unnecessary manual review rises. At 40 percent evidence loss, median gate latency increases to 95 minutes, showing both the protective value and operating cost of evidence-aware governance.

TABLE VII. SHIFTED-RISK ROBUSTNESS TEST

Model	AUC	F1	FN / FP
Static	0.624	0.477	49.6% / 30.1%
PDI	0.709	0.522	48.2% / 23.0%
Proposed	0.709	0.537	45.2% / 24.3%

Table VII lowers all predictive metrics because the test distribution differs from the original latent relationship. The PDI benchmark and the proposed framework remain above static governance, but the reduction confirms that simulator results are not production utility validation. Bootstrap intervals overlap: proposed ROC-AUC is 0.794 to 0.824 and PDI ROC-AUC is 0.789 to 0.820. The result supports an assurance-and-routing

interpretation of the proposed framework rather than a claim of large predictive superiority.

V. DISCUSSION AND THREATS TO VALIDITY

The results indicate that regulated utility release assurance is a layered decision problem rather than a single prediction task. Learned change, CI/CD, runtime, and governance signals improve discrimination, but prediction alone does not establish that a release is adequately evidenced, recoverable, policy-compliant, or appropriate for the current operating state. Deterministic routing supplies those control constraints, while the learned model supplies risk discrimination.

The evaluation focuses on software platforms supporting utility operations, including outage-management applications, customer portals, distributed-energy-resource coordination services, operational dashboards, market data interfaces, and cloud-hosted analytics. Demand, forecast error, ramp, generation, interchange, and volatility provide operating-state context. They do not replace software-engineering signals or serve as failure labels; they alter release tolerance when operating conditions are more sensitive.

The primary limitation is that software-release records are synthetic. The utility operating context is real, but release events, CI/CD signals, evidence states, rollback states, and failure labels are generated. The reported results should therefore be treated as a reproducible experimental evaluation of architecture and governance behavior, not proof of production utility outcomes or observed failure rates.

The second limitation is simulator external validity. The latent risk function includes change complexity, CI instability, runtime health, evidence weakness, application criticality, and utility context. Several of these signal families are later evaluated by the models. The shifted-risk test, ablation analysis, and policy-only baseline reduce this concern but do not eliminate dependence on simulator design. Real utility validation remains necessary.

The third limitation is the available utility-context extract. The primary ISO-NE file covers 4,417 hourly observations from 07-01-2021 through 01-01-2022. A longer extract would improve seasonal coverage. Utility operating sensitivity is also a proxy: high load, forecast error, ramp rate, or volatility may increase operational sensitivity, but they do not automatically mean a software release is unsafe. The secondary Kaggle dataset remains useful for robustness profiling, but deterministic derived fields prevent naive supervised-label use.

The optional real CI sanity-check protocol is not a substitute for utility validation. TravisTorrent, Rausch-derived build-failure data, and Huang replication materials can support feature-level plausibility [6], [8], [13], but they lack utility deployment approvals, rollback outcomes, operating-context decisions, and post-release incident records.

VI. CONCLUSION

This study developed and evaluated an evidence-aware predictive delivery and continuous software-assurance framework for regulated energy utility platforms. The framework integrates a multi-signal release-risk estimate with machine-verifiable evidence completeness, rollback readiness, mandatory policy state, utility operating sensitivity, and deterministic risk-tiered routing. Real ISO-NE observations derived from EIA-930 provide the operating context, while reproducible synthetic release records supply software-delivery signals and outcomes that are unavailable in public utility

datasets. The architecture therefore connects software risk and utility context without treating grid observations as software-failure labels.

The empirical results show that learned multi-signal release-risk modeling improves discrimination over static governance. The proposed framework reaches ROC-AUC 0.810 and F1 0.602; the external predictive-delivery benchmark [5] reaches 0.805 and 0.600. Evidence ablation produces 68.6 percent policy escape, while the full framework maintains 0.0 percent through deterministic enforcement. Routing becomes more conservative as utility operating sensitivity rises, placing the principal contribution in evidence-aware, context-sensitive assurance rather than marginal predictive lift.

Real utility CI/CD validation remains necessary before the framework can be claimed as validated for production utility delivery. Future work should replace synthetic release telemetry with de-identified CI/CD, deployment, incident, rollback, and evidence records from operating utilities. A shadow-mode validation should score real releases and produce recommended routes without authorizing production deployment. Such a study should measure calibration under actual failure labels, route accuracy, operator trust, override behavior, post-release health, evidence-generation reliability, and whether context-aware routing improves decisions during both normal and operationally sensitive periods.

REFERENCES

- [1] U.S. Energy Information Administration, "Hourly demand, day-ahead demand forecast, net generation, and interchange by balancing authority," Form EIA-930, Hourly Electric Grid Monitor. [Online]. Available: <https://www.eia.gov/opendata/browser/electricity/rto>. Accessed: Jun. 17, 2026.
- [2] North American Electric Reliability Corporation, "CIP-010-4: Cyber Security - Configuration Change Management and Vulnerability Assessments," NERC Reliability Standard, Sep. 14, 2022. [Online]. Available: <https://www.nerc.com/standards/reliability-standards/cip/cip-010-4>. Accessed: Jun. 17, 2026.
- [3] North American Electric Reliability Corporation, "CIP-013-2: Cyber Security - Supply Chain Risk Management," NERC Reliability Standard, 2020. [Online]. Available: <https://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx>. Accessed: Jun. 17, 2026.
- [4] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A large-scale empirical study of just-in-time quality assurance," *IEEE Trans. Softw. Eng.*, vol. 39, no. 6, pp. 757-773, Jun. 2013, doi: 10.1109/TSE.2012.70.
- [5] A. D. Agade and S. Balpande, "Predictive Delivery Intelligence for Payments and Core Banking: Reducing Change Risk at Scale," *Journal of Advances in Developmental Research*, vol. 15, no. 1, Jan.-Jun. 2024.
- [6] Y. A. Huang, D. A. da Costa, G. Dick, M. El Mezouar, and L. Xiao, "Is this build failure related to my patch? An empirical study of unrelated build failures in continuous integration," *Empirical Software Engineering*, vol. 31, no. 6, art. 148, 2026, doi: 10.1007/s10664-026-10874-8.
- [7] M. Souppaya, K. Scarfone, and D. Dodson, Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities, NIST SP 800-218, Feb. 2022, doi: 10.6028/NIST.SP.800-218.
- [8] T. Rausch, W. Hummer, P. Leitner, and S. Schulte, "An empirical analysis of build failures in the continuous integration workflows of Java-based open-source software," in *Proc. IEEE/ACM 14th Int. Conf. Mining Software Repositories (MSR)*, 2017, pp. 345-355, doi: 10.1109/MSR.2017.54.
- [9] K. Stouffer et al., Guide to Operational Technology (OT) Security, NIST SP 800-82 Rev. 3, Sep. 2023, doi: 10.6028/NIST.SP.800-82r3.
- [10] OpenSSF SLSA Community, "SLSA specification, version 1.1," Apr. 2025. [Online]. Available: <https://slsa.dev/spec/v1.1/>. Accessed: Jun. 17, 2026.
- [11] A. D. Agade and S. Balpande, "Platform Engineering in Regulated Finance: Self-Service Without Losing Control," *International Journal of Innovative Research and Creative Technology*, vol. 11, no. 3, 2025.
- [12] National Cybersecurity Center of Excellence, Situational Awareness for Electric Utilities, NIST SP 1800-7, Aug. 2019, doi: 10.6028/NIST.SP.1800-7.
- [13] M. Beller, G. Gousios, and A. Zaidman, "TravisTorrent: Synthesizing Travis CI and GitHub for full-stack research on continuous integration," in *Proc. IEEE/ACM 14th Int. Conf. Mining Software Repositories (MSR)*, 2017, pp. 447-450, doi: 10.1109/MSR.2017.24.
- [14] DatasetEngineer, "ISONE Smart City Energy Dataset," Kaggle, 2025, doi: 10.34740/KAGGLE/DSV/11806658.
- [15] A. D. Agade and S. Balpande, "A Reliability Control Plane for Regulated CI/CD: Integrating On-Call Triage, Risk-Tiered Release Governance, and Evidence-by-Design," *IJRMPS*, vol. 12, no. 5, Sep.-Oct. 2024.
- [16] U.S. Department of Energy, Office of Cybersecurity, Energy Security, and Emergency Response, "DOE OE-417 annual summaries," Open Energy Data Portal. [Online]. Available: <https://openenergyhub.ornl.gov/explore/dataset/oe-417-annual-summaries/>. Accessed: Jun. 17, 2026.