International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Telemetry-Native Reliability Engineering for Agentic AI Systems: Topology-Aware Failure Correlation and Decision Traceability

Shivam Dube[1], Rajat Varshney[2], Sanjay Kumar[3]

[1,2,3]Independent Researcher, School of Computer Science and Engineering, Galgotias University, Greater Noida, India
[1]shivam.dube21scse1022003@galgotiasuniversity.edu.in, [2]rajat.varshney20scse1022104@galgotiasuniversity.edu.in, [3]sanjay.kumar17scse1008765@galgotiasuniversity.edu.in;

Abstract
Agentic artificial intelligence systems combine large language models, tool-calling runtimes, retrieval services, application programming interfaces, policy guardrails, memory stores, and human approval paths. These systems create a reliability problem that differs from conventional microservices because failures may arise from model drift, tool misuse, dependency latency, invalid action plans, retrieval errors, policy denials, or release changes across the agent execution graph. This paper proposes Telemetry-Native Reliability with Decision Traceability (TNR-DT), a framework that combines OpenTelemetry-aligned signal normalization, topology-aware incident correlation, predictive change-risk scoring, deterministic policy routing, rollback-readiness checks, and decision-trace evidence generation. The study evaluates the framework through a benchmark-referenced reproducible simulation. It uses current public benchmark references, including ITBench, ITBench Trajectories, AIOps2025/RCA100, AgentOps-Bench, OpenTelemetry Demo Dataset, and GH Archive, to define operational scenarios, telemetry classes, fault categories, and change-event features. A fixed-seed AgentOps replay then evaluates threshold-local alerting, global time-window grouping, and TNR-DT across 100 seeds and 240 incident episodes per seed. TNR-DT reduced non-actionable alerts per incident from 8.69 to 3.11, reduced modeled mean time to restore from 101.81 to 54.23 minutes, improved top-three root-cause hit rate from 55.07% to 81.30%, increased evidence completeness from 53.00% to 96.08%, and reduced policy-exception escape rate from 11.30% to 2.57%. Change-risk prediction improved only modestly, with area under the curve increasing from 0.620 to 0.653. The stronger contribution is therefore not predictive superiority alone, but a traceable reliability operating model that links agent telemetry, dependency topology, policy decisions, rollback evidence, and incident reconstruction.

Keywords: Agentic AI, AIOps, Telemetry Correlation, Reliability Engineering, Decision Traceability, Policy-as-Code.

1. Introduction

Large language model-based autonomous agents increasingly execute multi-step workflows that plan, call tools, retrieve context, update memory, and interact with external systems. Recent surveys describe agents as systems that combine planning, memory, action, and environment interaction rather than isolated

prompt-response models [1]. As these agents enter information technology automation, security operations, financial operations, customer support, and software engineering workflows, reliability evaluation must shift from simple task completion to operational correctness, recovery, and evidence. ITBench illustrates this shift by benchmarking AI agents on real-world inspired Site Reliability Engineering (SRE), Compliance and Security Operations (CISO), and Financial Operations (FinOps) tasks [2].

Agentic AI reliability also has a governance dimension. The NIST Generative AI Profile highlights risks that include unreliable outputs, unsafe action, opaque behavior, and weak accountability mechanisms [3]. These concerns are amplified when an agent can open a ticket, modify a configuration, call a deployment tool, change a policy, retrieve sensitive context, or recommend a remediation. In these settings, reliability is not only a model-quality issue. It is a systems-engineering problem involving telemetry, dependency mapping, change risk, policy boundaries, rollback readiness, and incident evidence.

Conventional observability is necessary but incomplete. A distributed trace may show that an API call failed or that a service became slow, but it may not explain why an agent selected that tool, whether a retrieval miss influenced the plan, whether a policy denial was handled correctly, or whether a recent prompt or schema change contributed to the incident. Prior work on correlated telemetry in high-volume banking systems showed that operational signals become more useful when they are normalized, mapped to dependency context, compressed into incident objects, and linked to ranked root-cause candidates [4]. This paper extends that idea to agentic AI systems, where the dependency graph includes agent roles, models, prompts, tools, APIs, retrieval stores, policy modules, deployment artifacts, and human approval gates.

This study evaluates the framework through a benchmark-referenced reproducible simulation and architecture analysis. Production deployment outcomes, field-measured enterprise savings, and longitudinal operational effects are outside the scope of the present evaluation. Public 2024-2026 dataset and benchmark references are used to define scenario families and schema expectations, while a fixed-seed AgentOps replay creates experimental events for baseline comparison, ablation, stress testing, and robustness testing. The study asks four research questions. RQ1: Can topology-aware correlation reduce non-actionable alerts in agentic AI incidents compared with threshold-local and global time-window baselines? RQ2: Does predictive change-risk scoring materially improve incident prioritization, or is its value secondary to governance and evidence controls? RQ3: Can deterministic policy routing and rollback checks improve evidence completeness and reduce unsafe exception handling? RQ4: How stable is the framework under burst load, tool outage, degraded scoring, missing topology, noisy traces, and unseen tool types?

2. Literature Review

2.1. Agentic AI Operations and Observability

Agentic AI systems extend conventional software services because they reason over goals, produce intermediate plans, select external tools, and adapt behavior across steps. OpenTelemetry provides a useful foundation for metrics, logs, traces, and semantic conventions in distributed systems [5], and W3C Trace Context standardizes propagation of trace identifiers across service boundaries [6]. These standards support correlation, but agentic systems require additional artifacts such as prompt version, tool schema, policy decision, retrieval index version, and plan transition state. AgentOps-Bench addresses part of this

problem by evaluating tool-using LLM agents on completion, cost, efficiency, reliability, recovery, and safety under injected tool failures such as timeouts and malformed responses [7].

Agent observability requires a lifecycle view. The system must capture not only whether a request succeeded, but also which plan was formed, which tool was selected, which context was retrieved, what exception occurred, and how the agent responded. The telemetry-native premise of TNR-DT is that agent behavior should be represented as operational evidence from the start, rather than reconstructed after an incident from unrelated logs.

2.2. Change-Risk Scoring and Delivery Evidence

Predictive Delivery Intelligence introduced a release-risk model that combines change-centric, pipeline-centric, and runtime-centric signals with evidence bundles and risk-aware deployment gates [8]. TNR-DT adapts that model to agentic AI components. A change to a prompt template, tool schema, retrieval index, model-routing rule, orchestration function, or policy module may alter agent behavior even when the surrounding infrastructure remains healthy. GH Archive provides a public event stream for software-change activity and can support reproducible extraction windows for change-event modeling [9]. Runtime observability can be anchored through production-like datasets such as the OpenTelemetry Demo Dataset, which packages service telemetry around a demo application environment [10].

The present framework uses risk scoring as a supporting input, not as the central claim. In regulated or high-assurance environments, a modest but interpretable risk score may be more useful than a higher-performing opaque model if it can explain the signal family, route the decision, trigger rollback checks, and generate reviewable evidence.

2.3. Failure Diagnosis Benchmarks and Policy-Bounded Remediation

Recent failure-diagnosis benchmarks are moving toward multimodal observability and reasoning-process evaluation. AIOps2025 and RCA100 provide expert-labeled failure cases across representative microservice systems with metrics, logs, traces, events, alerts, topology, and causal evidence [11]. These datasets are directly relevant to agentic failure diagnosis because an agent must ground its answer in the evidence, not only name a faulty component. However, microservice root-cause analysis is not the same as agentic AI governance. Agent systems also need policy decisions, action approval, redacted evidence, and rollback verification.

Policy-as-code mechanisms provide a practical way to separate advice from execution. Open Policy Agent (OPA) uses Rego to evaluate structured input and return decisions [12]. Evidence-first autonomous remediation in regulated DevOps emphasizes policy decisions, GitOps execution, post-action validation, and audit-grade proof as part of the remediation loop [13]. TNR-DT uses that principle to ensure that agent-recommended actions remain bounded by explicit rules and that every allow, amend, deny, or escalate decision is recorded.

Secure delivery and AI lifecycle controls also matter. NIST SP 800-218A extends secure software development practices for generative AI and dual-use foundation models [14]. SLSA describes supply-chain provenance levels that support evidence about what was built and how [15]. In regulated CI/CD, a relia-

bility control plane connects on-call triage, risk-tiered governance, exception handling, rollback validation, and evidence-by-design [16]. TNR-DT generalizes this control-plane logic to agentic AI operations by placing policy routing and rollback readiness between prediction and execution.

2.4. Agent Trajectories and Reliability Stress

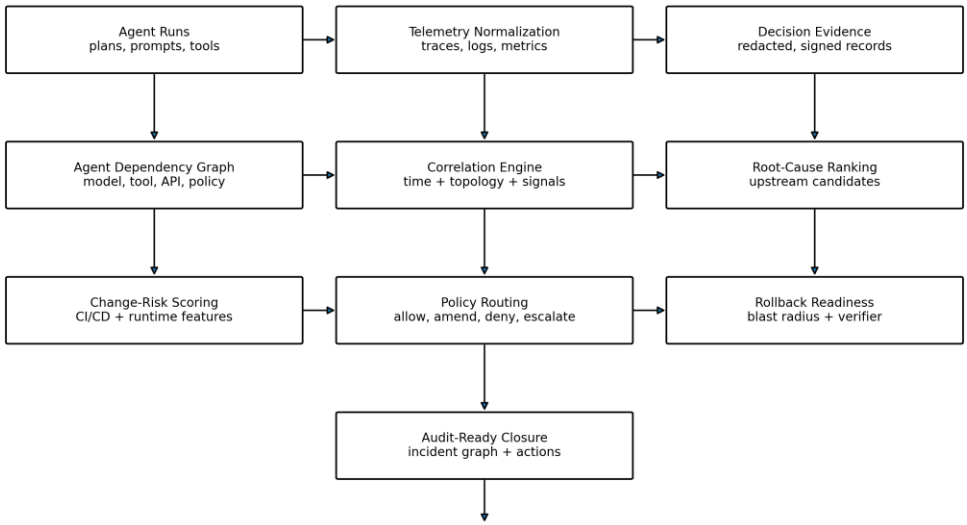
ITBench Trajectories provides released agent execution traces for SRE scenarios, including investigation workflow, generated code, final output, evaluation metrics, alerts, events, metrics, traces, logs, and Kubernetes object states [17]. ReliabilityBench evaluates LLM-agent reliability under repeated execution, semantic perturbation, and tool or API failure conditions [18]. These references support the stress-testing portion of the present study. Multi-agent systems such as AutoGen and ReAct-style agents show how agents can coordinate tool use and reasoning steps, but they also introduce failure surfaces that static service monitoring does not fully capture [19], [20].

3. Methodology

3.1. Architecture

Telemetry-Native Reliability with Decision Traceability (TNR-DT) is designed for agentic AI systems deployed in cloud, enterprise, or regulated digital environments. The architecture contains six functional layers: telemetry normalization, agent dependency graph construction, topology-aware incident correlation, predictive change-risk scoring, deterministic policy routing, and decision-trace evidence generation. Figure 1 summarizes the architecture.

Figure 1: TNR-DT Reference Architecture for Agentic AI Reliability



The system receives traces, logs, metrics, model events, tool-call events, retrieval events, deployment records, and policy decisions. Each event is normalized into a common representation: $x = \{id, t, c, s, f, a, p, r\}$, where id is the event identifier, t is timestamp, c is component identity, s is signal family, f is feature vector, a is action context, p is policy context, and r is redaction state. Redaction state is included because

agent traces may contain prompts, retrieved content, or tool outputs that cannot be stored as raw incident evidence.

The agent dependency graph is represented as G = (V, E, T), where V is the set of components, E is the set of directed dependencies, and T maps each component to a type. Component types include agent role, large language model endpoint, prompt template, tool connector, external API, retrieval index, memory store, orchestration state, policy module, runtime service, deployment artifact, and human approval gate. This graph converts an agent run into an operational topology that can support incident grouping and root-cause ranking.

Table 1: Telemetry Signal Families and Agentic AI Features

Signal family	Agentic AI examples	Reliability use
Model telemetry	model latency, token count, timeout, retry, refusal, routing decision	detects model degradation and routing instability
Tool telemetry	tool name, schema validity, exception type, API status, retry count	detects tool misuse and connector failure
Retrieval telemetry	retrieval latency, empty result rate, index version, embedding model version	detects stale or degraded retrieval
Policy telemetry	allow, deny, amend, escalation reason, approval status	proves controlled decision routing
Runtime telemetry	CPU, memory, queue depth, pod restart, network error, saturation	detects infrastructure contribution
Delivery telemetry	commit, artifact version, prompt version, tool-schema change, deployment window	supports change-risk scoring
Evidence telemetry	trace completeness, policy record, rollback check, verifier result	supports audit-ready reconstruction

3.2. Topology-Aware Correlation

TNR-DT correlates events using temporal proximity, graph proximity, signal-family agreement, and change-fingerprint similarity. For two events xi and xj, the correlation score is:

S(xi, xj) = lambda_t exp(-Delta t / tau) + lambda_g / (1 + dG(ci, cj)) + lambda_s J(si, sj) + lambda_c C(ci, cj). (1)

In (1), Delta t is time separation, tau is a decay constant, dG is shortest-path distance in the agent dependency graph, J is signal-family similarity, and C is a change-fingerprint match. Events are grouped into an

incident candidate when S(xi, xj) is greater than or equal to theta. This adapts correlated telemetry principles to agent graphs and prevents unrelated events from merging only because they occur near the same time [4].

3.3. Predictive Change-Risk Scoring

The change-risk score follows an interpretable signal-family design rather than a black-box predictive objective. For an agent component v, risk is computed as:

Prisk(v) = sigma(beta0 + beta1 Xchange + beta2 Xpipeline + beta3 Xruntime + beta4 Xpolicy + beta5 Xagent). (2)

Here, Xchange includes prompt, schema, retrieval, policy, and orchestration changes. Xpipeline includes build status, deployment age, test instability, and rollback-test status. Xruntime includes service-level objective burn, latency drift, saturation, and dependency error rate. Xpolicy includes prior denials, exception frequency, and missing approvals. Xagent includes invalid tool calls, repeated planning loops, tool-switching rate, and retrieval misses. The PDI signal-family concept is used here because it links release risk to traceable evidence and governance decisions rather than treating prediction as a standalone classifier [8].

3.4. Root-Cause Ranking and Decision Traceability

For each incident cluster C, candidate component v receives a root-cause score:

R(v, C) = w1 Ev + w2 Bv + w3 Prisk(v) + w4 Dv - w5 Yv. (3)

Ev is early-signal strength, Bv is graph position, Prisk(v) is the change-risk score, Dv is signal diversity, and Yv is a downstream-symptom penalty. A model gateway that shows the earliest latency drift and is upstream of repeated tool failures receives a higher score than a downstream tool that merely reports timeouts after the cascade begins. The output is a ranked candidate list, not a final adjudication of causality.

Decision traceability is represented by an evidence completeness score:

EC = |Apresent| / |Arequired|. (4)

Required artifacts include the incident cluster identifier, dependency subgraph, root-cause ranking explanation, risk-score feature summary, policy decision record, human approval record when applicable, action or non-action rationale, rollback-readiness check, post-action verification, and redaction metadata. The evidence-first design draws from policy-bounded remediation patterns where each action must be linked to a policy decision, execution artifact, and post-action validation [13].

3.5. Policy Routing and Rollback Controls

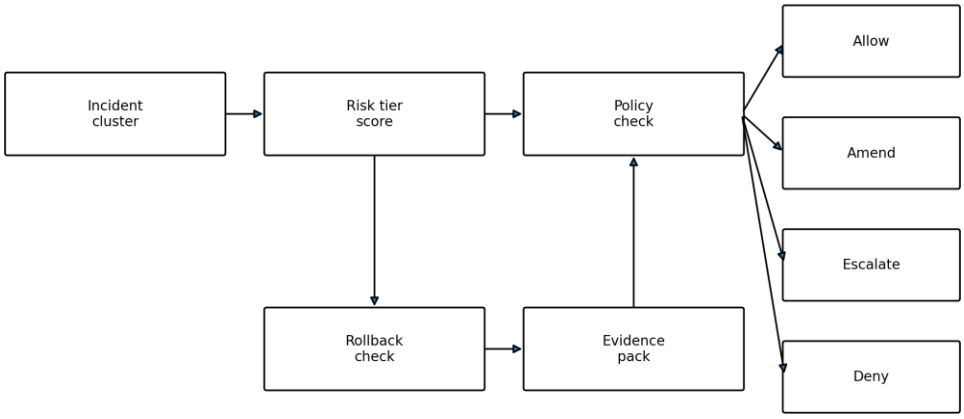
TNR-DT treats policy routing as deterministic control logic. The routing function is:

Decision = Pi(A, Tr, Ec, Rb, H). (5)

A is the proposed action, Tr is risk tier, Ec is evidence completeness, Rb is rollback readiness, and H is human approval status. The output is allow, amend, deny, or escalate. This makes prediction subordinate to policy. A high risk score may trigger additional validation, but it cannot authorize a privileged action by itself. Low-risk actions may proceed when allowlisted and reversible. Medium-risk actions may require constrained execution or human review. High-risk and emergency actions require explicit approval, narrower blast radius, or advisory-only handling. This design follows the reliability control-plane principle that on-call triage, release governance, exception handling, rollback validation, and evidence capture should operate as one control system [16].

Implementation can use Kubernetes NetworkPolicy for runtime network boundaries [21], Argo CD sync phases and waves for ordered GitOps reconciliation [22], and Sigstore Cosign for signed artifacts and attestations [23]. These tools support implementation, while the evaluated contribution lies in the integrated reliability loop connecting telemetry, topology, risk scoring, deterministic routing, rollback readiness, and evidence generation for agentic AI systems.

Figure 2: Deterministic Policy Routing and Evidence Closure Behavior



Routing decisions are deterministic controls. Prediction supplies a signal, but policy and rollback checks decide execution.

4. Experimental Design

The study uses a benchmark-referenced synthetic replay. Public datasets and benchmarks from 2024 to 2026 are used to define task classes, telemetry modalities, fault categories, and agent-operation artifacts. The generated replay then provides controlled conditions for baseline comparison, ablation, stress testing, and robustness testing. The design intentionally distinguishes source references from generated measurements. The results are simulation outputs produced by the fixed replay parameters, not direct measurements from a production agent platform.

Table 2: Dataset and Benchmark Use

Source	**Data type**	**Use in this paper**	**Limitation**
ITBench, 2025 [2]	SRE, CISO, and FinOps agent scenarios	Benchmark taxonomy, operational task classes,	Not used as production incident data
ITBench Trajectories, 2025 [17]	Agent reasoning, tool use, logs, traces, metrics,	Trace artifact schema and RCA evaluation	Coverage is limited to released SRE trajecto-
AIOps2025 and RCA100, 2026 [11]	Expert-labeled micro-service failure cases	Fault-template and RCA ground-truth design	Microservice RCA, not full agent governance te-
AgentOps-Bench, 2026 [7]	Tool-using LLM-agent tasks with injected tool failures	Stress conditions, recovery, tool failure classes	Not specific to regulated change governance

Source	Data type	Use in this paper	Limitation
OpenTelemetry Demo Dataset, current [10]	Production-like service telemetry from a demo stack	Runtime topology and telemetry replay design	Demo workload, not enterprise production
GH Archive, 2024-2026 extraction window [9]	Public GitHub event stream	Change-event features for release-risk model-	Open-source activity is an imperfect proxy for

The replay creates agent incident episodes across model, tool, API, retrieval, policy, runtime, and deployment components. Each episode has an originating component, dependency neighborhood, signal timeline, change-adjacency flag, severity label, recovery path, policy decision path, rollback-readiness state, and generated evidence artifacts. The replay uses 100 seed-level replications and 240 incidents per seed. Table 3 summarizes fixed parameters.

Table 3: Fixed Replay Parameters

Parameter	Value	Rationale
Replay seeds	100	Seeds 1000-1099 for primary comparison
Incidents per seed	240	Synthetic agent incident episodes
Observation window	90 days	Virtual replay horizon
Component types	8	Agent, model, tool, API, retrieval, policy, runtime, deployment
Correlation window	8 minutes	Consistent with short-burst operational triage
Graph-hop limit	2 hops	Prevents unrelated simultaneous events from merging
Risk tiers	low, medium, high, emergency	Used for policy routing and rollback requirements
Confidence intervals	95%	Computed across seed-level results

Three configurations are evaluated. Baseline A uses threshold-local alerting with local duplicate suppression. Baseline B adds global time-window grouping but does not use agent dependency topology or policy-aware routing. TNR-DT uses topology-aware correlation, change-risk scoring, deterministic policy routing, rollback-readiness checks, and decision-trace evidence generation. Metrics include non-actionable alerts per incident, median detection latency, mean time to restore (MTTR), top-one and top-three root-

cause hit rate, change-risk area under the curve (AUC), evidence completeness, policy-exception escape rate, rollback-readiness rate, and routing error rate.

The simulation uses fixed random seeds and reports mean values with 95% confidence intervals across seed-level outputs. AUC is computed on change-adjacent incident labels within the replay. Evidence completeness and policy exceptions are deterministic-control outcomes, while RCA hit rate and MTTR combine stochastic incident generation with control-path effects. This separation is necessary because predictive effects and deterministic governance effects have different technical implications. Section 9 documents the replay seed policy, event schema, reproducibility files, and ablation configurations used to support independent replication of the benchmark-referenced simulation.

Table 4: Experimental Configurations

Capability	**Baseline A**	**Baseline B**	**TNR-DT**
Per-component thresholds	Yes	Yes	Yes
Local deduplication	Yes	Yes	Yes
Global time-window grouping	No	Yes	Yes
Agent dependency graph	No	No	Yes
Topology-aware root-cause ranking	No	No	Yes
Change-risk scoring	No	Partial	Yes
Deterministic policy routing	No	No	Yes
Rollback-readiness check	Partial	Partial	Yes
Generated decision-trace evidence	No	Partial	Yes

5. Results

5.1. Baseline Comparison

Table 5 presents the primary baseline comparison. TNR-DT reduced non-actionable alerts per incident from 8.69 under threshold-local alerting and 5.33 under global time-window grouping to 3.11. Mean MTTR declined from 101.81 minutes under Baseline A to 54.23 minutes under TNR-DT. Top-three root-cause hit rate improved from 55.07% to 81.30%. Evidence completeness increased from 53.00% to 96.08%, and policy-exception escape rate decreased from 11.30% to 2.57%.

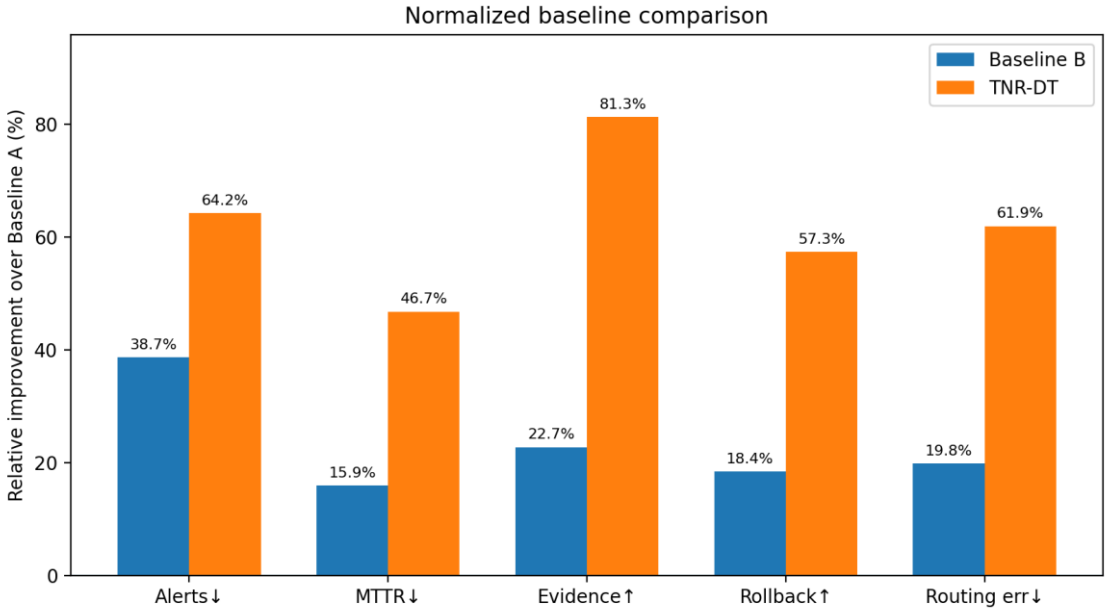
The change-risk AUC improved from 0.620 to 0.653. This is a modest predictive improvement. The result supports using change-risk scoring as a routing and prioritization signal, but it does not support a claim that prediction alone explains the framework outcome. The larger improvements come from telemetry correlation, deterministic routing, rollback checks, and evidence generation.

Table 5: Primary Results, Mean ± 95% Confidence Interval

Metric	**Baseline A**	**Baseline B**	**TNR-DT**
Non-actionable alerts/incident	8.69 ± 0.05	5.33 ± 0.03	3.11 ± 0.02
Median detection latency (min)	20.85 ± 0.06	19.36 ± 0.06	16.33 ± 0.05
Mean MTTR (min)	101.81 ± 0.34	85.62 ± 0.37	54.23 ± 0.38
RCA top-one accuracy (%)	30.46 ± 0.58	43.44 ± 0.66	60.74 ± 0.65
RCA top-three hit rate (%)	55.07 ± 0.59	64.95 ± 0.61	81.30 ± 0.54
Change-risk AUC	0.620 ± 0.009	0.627 ± 0.008	0.653 ± 0.008
Evidence completeness (%)	53.00 ± 0.13	65.03 ± 0.14	96.08 ± 0.04
Policy-exception escape rate (%)	11.30 ± 0.38	8.80 ± 0.40	2.57 ± 0.19
Rollback-readiness rate (%)	57.82 ± 0.57	68.45 ± 0.63	90.97 ± 0.40

Metric	Baseline A	Baseline B	TNR-DT
Routing error rate (%)	28.80 ± 0.56	23.10 ± 0.45	10.96 ± 0.37

Figure 3: Relative Improvement Over Threshold-Local Baseline Across Alert, MTTR, Evidence, Rollback, and Routing Metrics



5.2. Ablation Results

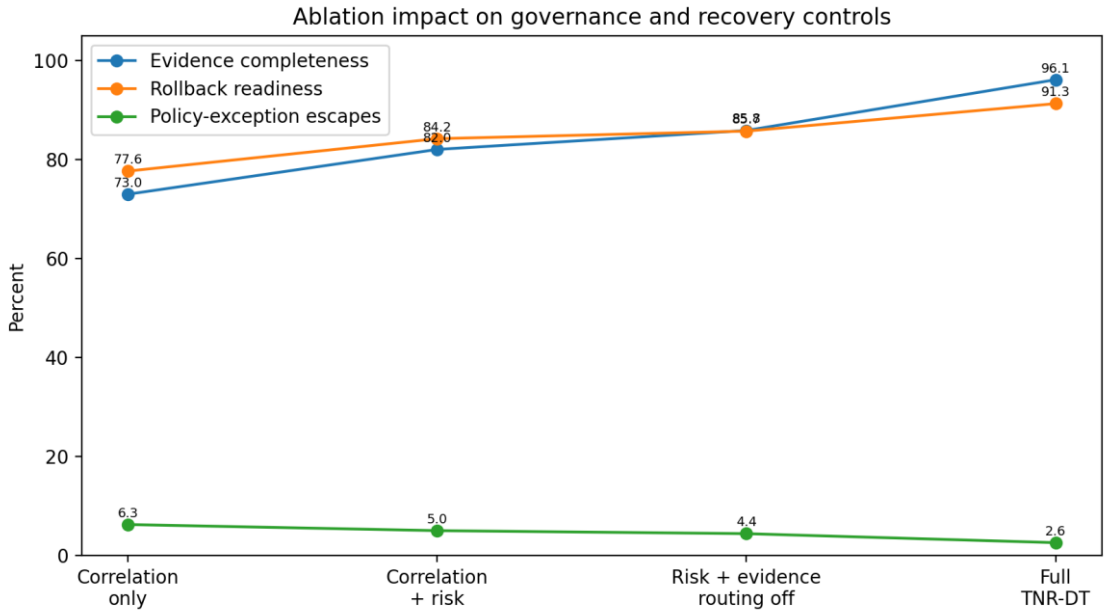
The ablation study separates correlation, change-risk scoring, generated evidence, and full control-plane routing. The variant labeled risk + evidence, routing off keeps the evidence artifacts but removes deterministic allow, amend, deny, and escalate routing derived from reliability control-plane logic [16]. Correlation-only reduces alert volume because it groups symptoms into incident objects, but its evidence completeness remains 72.97% and policy-exception escape rate remains 6.26%. Adding change-risk scoring improves MTTR and root-cause ranking. Adding evidence without routing improves artifact coverage but leaves more policy exceptions and lower rollback readiness than full TNR-DT. Full TNR-DT raises evidence completeness to 96.09%, lowers policy-exception escapes to 2.60%, and raises rollback readiness to 91.26%.

Table 6: Ablation Results, Mean ± 95% Confidence Interval

Metric	Corr. only	Corr. + risk	Risk + Evidence	Full TNR-DT
Non-actionable alerts/incident	3.10 ± 0.02	3.11 ± 0.01	3.10 ± 0.01	3.09 ± 0.01

Metric	Corr. only	Corr. + risk	Risk + Evidence	Full TNR-DT
Mean MTTR (min)	67.31 ± 0.34	60.67 ± 0.30	58.92 ± 0.32	54.13 ± 0.34
RCA top-three hit rate (%)	73.56 ± 0.49	79.12 ± 0.54	80.21 ± 0.50	81.77 ± 0.48
Change-risk AUC	0.615 ± 0.008	0.655 ± 0.008	0.655 ± 0.008	0.655 ± 0.008
Evidence com-pleteness (%)	72.97 ± 0.11	82.01 ± 0.10	85.84 ± 0.08	96.09 ± 0.04
Policy-exception escape rate (%)	6.26 ± 0.32	5.03 ± 0.26	4.42 ± 0.24	2.60 ± 0.19
Rollback-readiness	77.64 ± 0.52	84.18 ± 0.55	85.71 ± 0.46	91.26 ± 0.38

Figure 4: Ablation Impact on Evidence Completeness, Rollback Readiness, and Policy Exceptions



5.3. Stress and Robustness Results

Stress testing shows that burst load is the most visible operational stressor. Under burst x2, non-actionable alerts rise from 3.10 to 5.87 and MTTR rises from 54.33 to 62.09 minutes. Tool outage also increases MTTR to 59.79 minutes. Model degradation reduces change-risk AUC from 0.660 to 0.639. Evidence completeness and rollback readiness remain stable because they are generated by deterministic controls rather than learned predictions.

Table 7: Stress Testing Results, Mean ± 95% Confidence Interval

Metric	**Nominal**	**Burst x2**	**Tool outage**	**Model degradation**
Non-actionable alerts/incident	3.10 ± 0.02	5.87 ± 0.02	3.10 ± 0.02	3.10 ± 0.02
Mean MTTR (min)	54.33 ± 0.34	62.09 ± 0.32	59.79 ± 0.34	56.03 ± 0.34
RCA top-three hit rate (%)	82.04 ± 0.50	82.18 ± 0.51	82.04 ± 0.50	82.04 ± 0.50
Change-risk AUC	0.660 ± 0.008	0.650 ± 0.008	0.660 ± 0.008	0.639 ± 0.008
Evidence completeness (%)	96.09 ± 0.04	96.11 ± 0.03	96.09 ± 0.04	96.09 ± 0.04
Policy-exception escape rate (%)	2.45 ± 0.19	2.50 ± 0.19	2.45 ± 0.19	2.45 ± 0.19
Rollback-readiness rate (%)	90.95 ± 0.33	90.55 ± 0.36	90.95 ± 0.33	90.95 ± 0.33

Robustness testing shows that missing topology is more damaging than trace noise. When 15% of topology links are missing, non-actionable alerts increase to 3.29 and routing error rises to 11.40%. Unseen tool types slightly reduce evidence completeness and rollback readiness. This suggests that production implementation would require a maintained tool registry, topology reconciliation, and coverage tests for new tool connectors.

Table 8: Robustness Results, Mean ± 95% Confidence Interval

Metric	**Nominal**	**Noisy traces +20%**	**Missing topology 15%**	**Unseen tool type 20%**
Non-actionable alerts/incident	3.11 ± 0.01	3.11 ± 0.01	3.29 ± 0.02	3.11 ± 0.01
Mean MTTR (min)	54.27 ± 0.33	54.34 ± 0.33	54.46 ± 0.33	54.46 ± 0.33

Metric	Nominal	Noisy traces +20%	Missing topology 15%	Unseen tool type 20%
RCA top-three hit rate (%)	81.94 ± 0.49	81.53 ± 0.50	81.14 ± 0.49	81.30 ± 0.49
Change-risk AUC	0.655 ± 0.007	0.652 ± 0.007	0.655 ± 0.007	0.655 ± 0.007
Evidence completeness (%)	96.06 ± 0.04	96.06 ± 0.04	96.06 ± 0.04	95.73 ± 0.04
Rollback-readiness rate (%)	90.80 ± 0.38	90.80 ± 0.38	90.80 ± 0.38	90.42 ± 0.38
Routing error rate (%)	10.84 ± 0.44	10.84 ± 0.44	11.40 ± 0.45	11.32 ± 0.45

6. Discussion

The results support the view that agentic AI reliability should be treated as a combined telemetry, topology, governance, and evidence problem. Threshold-local alerting creates unnecessary alert volume because each component behaves as a separate alerting surface. Global time-window grouping reduces some noise but cannot reliably distinguish an upstream agent, model, or tool failure from downstream symptoms. TNR-DT improves this behavior by using agent dependency context to compress symptoms and rank root-cause candidates.

The predictive result should be interpreted carefully. The change-risk AUC gain is modest, and the confidence intervals show that prediction is a supporting mechanism rather than the main contribution. This distinction is important for operational adoption and technical review. If an organization adopts TNR-DT, it should not justify the system on the assumption that the risk model is highly accurate. The results indicate that even a modest interpretable risk signal can be useful when it informs routing, escalation, rollback testing, and evidence capture.

The governance and evidence results are stronger than the predictive result. Evidence completeness increases because required artifacts are generated as part of the workflow. Policy-exception escapes decrease because the policy function forces allow, amend, deny, or escalate outcomes. Rollback readiness improves because action routing depends on recovery evidence rather than a generic approval record. The ablation variant with routing disabled confirms that evidence generation alone is not equivalent to a reliability control plane; deterministic routing is needed to reduce exception escapes and improve rollback readiness [16]. These are deterministic control effects, not model-prediction effects.

TNR-DT is most suitable for high-assurance agentic AI use cases, including SRE assistants, IT automation agents, compliance operations agents, infrastructure-remediation copilots, software-delivery agents, and enterprise support agents. The framework is less necessary for low-risk conversational assistants that do not call operational tools, modify systems, or require audit-grade reconstruction.

7. Threats to Validity and Limitations

External validity is limited because the reported values come from a fixed-seed replay rather than a production deployment. Public benchmark references shape the study design, but they do not fully represent enterprise agentic AI operations. Real organizations may have different tool catalogs, model-routing policies, topology completeness, approval latency, incident severity distributions, and rollback procedures.
Synthetic-data circularity is another concern. The replay generates incidents using dependency neighborhoods, and the proposed system is designed to exploit dependency neighborhoods. This can overstate the benefit of topology-aware correlation. Production validation should compare candidate root-cause rankings against independently labeled incident outcomes and include incidents where topology is stale, incomplete, or misleading.

Dataset proxy limits also affect interpretation. ITBench, ITBench Trajectories, AIOps2025/RCA100, AgentOps-Bench, OpenTelemetry Demo Dataset, and GH Archive are useful references, but no single source provides a complete regulated agentic AI incident dataset with model events, tool calls, policy decisions, release metadata, rollback evidence, and human approval records. A stronger empirical record would require a de-identified production trace corpus or a shared benchmark that includes the full AgentOps control path.

Prediction remains limited in this study. The change-risk model shows only modest AUC improvement in the replay, and calibration, drift, and subgroup performance are not fully assessed. Future validation should report calibration curves, false-positive cost, risk-tier stability, model drift, and separate results by agent workflow type.

Human factors are under-modeled. The simulation does not fully represent operator trust, override behavior, fatigue, disagreement with agent recommendations, or the quality of human approvals. Shadow-mode pilots should measure override frequency, time to understand the evidence pack, disagreement rates, and post-incident reviewer confidence.

8. Conclusion

This paper proposed TNR-DT, a telemetry-native reliability engineering framework for agentic AI systems. The framework combines normalized telemetry, agent dependency graphs, topology-aware failure correlation, predictive change-risk scoring, deterministic policy routing, rollback-readiness checks, and decision-trace evidence generation. The design adapts correlated telemetry, predictive delivery intelligence, evidence-first remediation, and reliability-control-plane methods to the operational context of agentic AI systems.

In a reproducible simulation of 100 seeds and 240 incident episodes per seed, TNR-DT reduced alert noise, improved root-cause candidate quality, shortened modeled MTTR, increased evidence completeness, reduced policy-exception escapes, improved rollback readiness, and reduced routing error compared with

threshold-local and time-window baselines. The predictive improvement was modest, with AUC rising from 0.620 to 0.653. The stronger contribution was the integration of telemetry, topology, policy routing, rollback readiness, and evidence into one reliability operating model.

Real-world validation remains necessary. Future work should run TNR-DT in shadow mode against de-identified production agent traces, use independently labeled incident outcomes, measure calibration and drift, compare operator override behavior, validate rollback outcomes, and link incidents to actual release and tool-schema changes across longer observation windows.

9. Reproducibility Appendix

This appendix records the replay parameters, seed policy, generated event schema, and ablation configuration used for the benchmark-referenced evaluation. It is included to make the simulation assumptions explicit and to support independent replication of the reported comparisons. The appendix does not replace production validation; it identifies the configuration that should accompany any public repository, supplementary archive, or institutional replication package.

9.1. Replication Package Contents

A supplementary replication package accompanies the manuscript and contains a plain-text readme, replay configuration, seed list, event schema, ablation matrix, and metric definitions. The package is designed for deposit in a public repository or journal supplement so that the reported tables and figures can be independently regenerated from the same seed policy and configuration files.

Table 9: Reproducibility Package Contents

File or folder	Purpose	Required content
README.md	Explains how the simulation is organized and how to reproduce tables and figures	Study scope, dataset references, random seed range, run order, expected outputs, and known limitations
config/replay_config.yaml	Defines fixed replay settings	100 seeds, seeds 1000-1099, 240 incidents per seed, 90-day replay horizon, component types, correlation window, graph-hop limit, and risk tiers
config/ablation_matrix.csv	Defines baseline and ablation variants	Threshold-local baseline, global time-window baseline, correlation-only, correlation + risk, risk + evidence without routing, and full TNR-DT

File or folder	Purpose	Required content
schema/agentops_event_schema.json	Defines event-level fields used in generated telemetry	Event identifier, timestamp, component identity, signal family, feature vector, action context, policy context, and redaction state
metrics/metric_definitions.csv	Defines reported measurements	Alert volume, detection latency, MTTR, RCA hit rate, AUC, evidence completeness, policy-exception escape rate, rollback readiness, and routing error
outputs/	Stores reproduced result files	Seed-level metrics, aggregated confidence intervals, generated tables, and figure data files

9.2. Seed Policy and Replay Logic

Seed-level replication is used to reduce dependence on a single generated incident sequence. The primary comparison uses seeds 1000 through 1099. Each seed produces 240 incident episodes across a 90-day virtual observation window. Each incident receives an originating component, a dependency neighborhood, an ordered signal timeline, a severity label, a change-adjacency flag, a recovery path, a policy decision path, a rollback-readiness state, and a generated evidence-artifact set. The same seed list is used across all baselines so that differences between configurations are attributable to correlation, risk scoring, routing, and evidence controls rather than to different incident samples.

Table 10: Event Schema and Seed Logic

Element	Specification	Replication role
Seed range	1000-1099	Controls repeated sampling and allows the same incident episodes to be replayed across all configurations
Incident count	240 episodes per seed	Produces 24,000 total replay episodes for seed-level aggregation

Element	Specification	Replication role
Observation window	90 virtual days	Provides repeated release windows, incident bursts, and recovery cycles
Event tuple	x = {id, t, c, s, f, a, p, r}	Preserves event identity, timestamp, component, signal family, feature vector, action context, policy context, and redaction state
Component types	agent, model, tool, API, retrieval, policy, runtime, deployment	Defines the node classes used to construct the agent dependency graph
Correlation rule	S(xi, xj) >= theta, with theta = 0.62	Applies a fixed incident-grouping threshold across replay configurations that use correlation
Graph-hop limit	2 hops	Constrains event grouping to nearby agent dependency neighborhoods
Risk tiers	low, medium, high, emergency	Controls routing requirements, rollback checks, and evidence obligations

9.3. Generated Event Schema

Each generated event follows a shared schema so that baseline and proposed configurations evaluate the same evidence. The event identifier and timestamp support ordering. The component identity and signal family support correlation. The feature vector stores numerical and categorical indicators such as latency, retry count, policy status, deployment age, rollback-test status, or retrieval miss rate. Action context records a proposed remediation or routing step when applicable. Policy context records allow, amend, deny, or escalate decisions. Redaction state records whether prompt, retrieval, or tool-output content is stored as raw text, hashed reference, or redacted summary.

The reproducibility package should include the schema in JSON form and a small validation script that rejects events with missing identifiers, timestamps, component types, signal families, or policy states. This check is important because evidence completeness is a deterministic-control metric; it can be overstated if incomplete events are silently accepted.

9.4. Ablation Configuration Matrix

The ablation matrix is included to clarify which mechanism is being tested in each condition. This directly separates the effect of topology-aware correlation, change-risk scoring, generated evidence, and deterministic control-plane routing. The routing-disabled condition is especially important because it tests whether evidence generation alone is sufficient, or whether the risk-tiered allow, amend, deny, and escalate control logic contributes additional reliability value.

Table 11: Ablation Configuration Matrix

Configuration	Correlation	Change-risk scoring	Evidence generation	Policy routing	Purpose
Threshold-local baseline	No	No	No	No	Measures per-component alerting with local duplicate suppression
Global time-window baseline	Time only	Partial	Partial	No	Measures alert grouping without agent dependency topology or control-plane routing
Correlation only	Yes	No	Partial	No	Isolates the value of topology-aware alert compression and root-cause ranking
Correlation + risk	Yes	Yes	Partial	No	Tests whether PDI-style change-risk features improve prioritization and MTTR

Configuration	Correlation	Change-risk scoring	Evidence generation	Policy routing	Purpose
Risk + evidence, no routing	Yes	Yes	Yes	No	Tests whether evidence generation alone improves governance outcomes without deterministic routing
Full TNR-DT	Yes	Yes	Yes	Yes	Evaluates the integrated telemetry, topology, risk, routing, rollback, and evidence loop

9.5. Metric Reproduction Rules

For each seed, metrics are computed at the incident level and then aggregated across seeds. Non-actionable alerts per incident are counted after each configuration performs its suppression or correlation step. Mean time to restore is computed from incident start to modeled recovery closure. Root-cause hit rate is computed against the simulated originating component and dependency neighborhood. Change-risk area under the curve is computed only on change-adjacent incident labels. Evidence completeness is calculated as |Apresent| / |Arequired| for the applicable risk tier. Policy-exception escape rate counts cases where an action path proceeds without the required policy state. Rollback readiness counts incidents with a valid rollback check, bounded blast radius, and post-action verifier state.

9.6. Reproducibility Limits

The appendix supports computational reproducibility of the benchmark-referenced replay, but it does not establish external validity by itself. Independent reproduction should report whether the same seed policy and schema produce comparable directionality under alternative incident distributions, different graph densities, additional tool types, and different policy thresholds. A stronger validation record would require de-identified production agent traces, independently labeled root-cause outcomes, operator override records, and actual rollback results.

Reference

1. L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J.-R. Wen, "A survey on large language model based autonomous agents," arXiv:2308.11432, 2023, revised 2025.

2. S. Jha et al., "ITBench: Evaluating AI agents across diverse real-world IT automation tasks," in Proceedings of the 42nd International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 267, pp. 27134-27197, 2025.
3. National Institute of Standards and Technology, Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, NIST AI 600-1, 2024, doi: 10.6028/NIST.AI.600-1.
4. A. D. Agade and S. Balpande, "From Alert Floods to Action: Correlated Telemetry for High-Volume Banking Systems," International Journal of Artificial Intelligence, Data Science, and Machine Learning, vol. 6, no. 1, pp. 240-249, 2025, doi: 10.63282/3050-9262.IJAIDSML-V6I1P127.
5. OpenTelemetry Authors, "OpenTelemetry Specification 1.60.0," OpenTelemetry, 2026, accessed Sep. 7, 2026. https://opentelemetry.io/docs/specs/otel/
6. World Wide Web Consortium, "Trace Context," W3C Recommendation, Nov. 23, 2021, accessed Sep. 7, 2026. https://www.w3.org/TR/trace-context/
7. K. S. Srivastav, "AgentOps-Bench: A reproducible benchmark for operational evaluation of tool-using LLM agents," software release and pilot trace bundle, 2026, doi: 10.5281/zenodo.19875693.
8. 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.
9. I. Grigorik, "GH Archive: A public record of GitHub public events," GH Archive, 2024-2026 extraction window, accessed Sep. 7, 2026. https://www.gharchive.org/
10. OpenObserve Authors, "OpenTelemetry Demo Dataset," GitHub repository, 2024-2026, accessed Sep. 7, 2026. https://github.com/openobserve/opentelemetry-demo-dataset
11. Y. Cai et al., "A multi-dataset benchmark for evaluating LLM agents in microservice failure diagnosis," arXiv:2606.29193, 2026.
12. Open Policy Agent Authors, "Policy Language," Open Policy Agent Documentation, 2026, accessed Sep. 7, 2026. https://www.openpolicyagent.org/docs/policy-language
13. A. D. Agade and S. Balpande, "From Incident to Evidence: Autonomous AIOps for DORA-Ready Financial DevOps, Self-Healing Remediation with Traceability and Audit-Grade Proof," International Journal for Multidisciplinary Research, vol. 7, no. 6, Nov.-Dec. 2025.
14. National Institute of Standards and Technology, Secure Software Development Practices for Generative AI and Dual-Use Foundation Models: An SSDF Community Profile, NIST SP 800-218A, 2024, doi: 10.6028/NIST.SP.800-218A.
15. SLSA Framework Authors, "SLSA Specification v1.0," Supply-chain Levels for Software Artifacts, 2023, accessed Sep. 7, 2026. https://slsa.dev/spec/v1.0/
16. 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," IJIRMPS, vol. 12, no. 5, Sep.-Oct. 2024.
17. IBM Research, "ITBench Trajectories," Hugging Face dataset, 2025, accessed Sep. 7, 2026. https://huggingface.co/datasets/ibm-research/ITBench-Trajectories
18. A. Gupta, "ReliabilityBench: Evaluating LLM Agent Reliability Under Production-Like Stress Conditions," arXiv:2601.06112, 2026, doi: 10.48550/arXiv.2601.06112.

19. Q. Wu et al., "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," arXiv:2308.08155, 2023; conference version, 2024.
20. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proceedings of the International Conference on Learning Representations, 2023.
21. Kubernetes Authors, "NetworkPolicy," Kubernetes Documentation, 2026, accessed Sep. 7, 2026. https://kubernetes.io/docs/reference/kubernetes-api/networking-resources/network-policy-v1/
22. Argo CD Authors, "Sync Phases and Waves," Argo CD Documentation, 2026, accessed Sep. 7, 2026. https://argo-cd.readthedocs.io/en/latest/user-guide/sync-waves/
23. Sigstore Authors, "Sigstore Cosign documentation and signing overview," Sigstore Documentation, 2026, accessed Sep. 7, 2026. https://docs.sigstore.dev/cosign/