Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Strengthening U.S. Digital Infrastructure Through AI-Powered Cloud-Native Software Resilience and Secure Delivery Systems

Jawad Yaqoob Mir[1]✉ and Fawad Mir[2]
[1,2]*Master of Science in Information Technology, Washington University of Science and Technology (WUST), Alexandria, Virginia, USA*
Corresponding Author: Jawad Yaqoob Mir **E-mail**: jawadmir488@gmail.com

| ABSTRACT

The growing dependence of government, healthcare, finance, energy, communications, transportation, and other critical sectors on software-intensive services has made cloud-native resilience and secure software delivery important components of U.S. digital infrastructure. Microservices, containers, orchestration platforms, and continuous integration and delivery pipelines improve scalability and release velocity, but they also introduce complex service dependencies, fragmented telemetry, cascading failures, vulnerable software components, and software supply-chain integrity risks. This study develops the AI-Powered Cloud-Native Resilience and Secure Delivery Framework (AICR-SDF), a conceptual artifact that connects critical-service requirements, cloud-native infrastructure, observability, AI-assisted operational intelligence, secure software delivery, and policy-governed response. The framework is developed using Design Science Research Methodology supported by an integrative literature review. Its technical foundations draw on established reliability, resilience, software-delivery, network-analysis, cybersecurity, and governance concepts, including availability and reliability functions, service-level objectives and error budgets, burn-rate analysis, DORA software-delivery metrics, network centrality, likelihood–impact risk assessment, the Common Vulnerability Scoring System, Supply-chain Levels for Software Artifacts, the NIST Secure Software Development Framework, and resilience-loss analysis. Four qualitative scenarios demonstrate how the framework distinguishes normal deployment, service degradation, an unverified software artifact, and cascading microservice failure. Ex ante evaluation examines requirements coverage, internal consistency, standards alignment, scenario applicability, governance adequacy, and implementability. The study contributes an integrated and policy-aware conceptual foundation for the future prototyping and empirical evaluation of resilient and securely delivered cloud-native services.

| KEYWORDS

Artificial intelligence; cloud-native systems; digital infrastructure; software resilience; secure software delivery; DevSecOps; software supply chain; design science research

| ARTICLE INFORMATION

ACCEPTED: 15 June 2026 **PUBLISHED:** 15 August 2026 **DOI:** 10.32996/jcsts.2026.8.8.26

1. Introduction

Digital infrastructure is an integral part of government, economy and society in the United States (Li et al., 2022). More and more software platforms are required for public administration, healthcare delivery, financial transactions, energy coordination, communications, transportation, emergency response and commercial operations, and they must always be available. Today, the trustworthiness of these services depends not only on the physical data centers or network connectivity, but it also requires software designs, cloud systems, identity management systems, API services, shared libraries, automated delivery pipelines, and external digital services. Thus, one layer's disruption can affect organizations and users beyond the source of the disruption.

Cloud-native computing is a widely adopted architecture style for developing and running today's digital services (Ugwueze, 2024). Microservices, containers, automated orchestration, service meshes, declarative infrastructure, and continuous integration and continuous delivery pipelines are all common components of cloud-native environments. These practices enable software

components to be developed, deployed, updated, and scaled independently. These can help them to make resources more efficient, flexible deployment, portability, and recovery from stand-alone infrastructure failures. The same characteristics also introduce substantial complexity. A single user request may pass through an API gateway, authentication service, transaction service, data store, message queue, and third-party API before completion. Each component may have a different release schedule, configuration, resource demand, security posture, and failure mode. Containers and service instances may be created, terminated, or relocated dynamically (Kaur et al., 2022). Consequently, the system cannot be understood adequately by monitoring individual machines or applications in isolation.

In distributed microservice systems, cascading failure is one of the concerns (Tariq et al., 2026). A slow downstream service can cause latency on upstream services, timeout and retry storms, can drain connection pools, and can stress and overload otherwise healthy services. Tools like redundancy, autoscaling, circuit breakers, bulkheads, health checks, rate limiting and automated restart can be used to increase resilience, however, it varies by context. Retries can exacerbate the overload and scaling doesn't fix a broken software version. It is then important that recovery depends on the evidence of workload, topology, deployment history, dependency conditions and likely source of degradation. Observability provides this evidence through metrics, logs, traces, events, and topology information (Souza, 2024). Metrics describe latency, traffic, errors, resource saturation, and service-level performance. Logs provide contextual records, while distributed traces reveal how requests move through service boundaries. The scale and heterogeneity of these signals make manual analysis increasingly difficult, particularly during incidents involving many correlated alerts. Artificial intelligence for IT operations offers mechanisms for anomaly detection, failure prediction, alert correlation, root-cause analysis, and response recommendation. AI can examine relationships among telemetry, service dependencies, deployment events, and historical behavior (Devalla, 2025). Cloud-native resilience is also inseparable from secure software delivery (Ugwueze, 2024). Modern applications are assembled through complex software supply chains involving source repositories, third-party packages, container base images, build services, testing tools, artifact registries, deployment credentials, and orchestration platforms. DevSecOps and software supply-chain assurance are mitigating these concerns by providing software supply-chain tools for secure development practices, code and dependency analysis, vulnerability management, software bills of materials, artifact signing, attestations, provenance, and policy enforcement. But delivery security and runtime resilience often are handled by separate teams and tools. Pipeline controls can accept an artifact without the aid of post-deployment evidence, and monitoring systems can see degradation but not know if the deployed artifact had a trusted and approved build process.

There is a gap between AIOps and AI governance (Sayles, 2024). There is a lack of guidance regarding the interplay between probabilistic recommendations and deterministic security requirements, pre-authorized automation and human approval, and how they should all work together in one cloud-native decision architecture. This research aims at filling in the missing link among cloud-native resilience, AI-driven operational intelligence, secure software delivery, software supply-chain assurance, policy enforcement and accountable governance. It brings forward the AI-Powered Cloud-Native Resilience and Secure Delivery Framework (AICR-SDF), which has six interdependent layers of Critical Digital Services, Cloud-Native Infrastructure, Observability & Evidence, AI Resilience Intelligence, Secure Software Delivery, and Policy, Response & Governance. The research has four contributions. First, it proposes an integrated multilayer framework that combines operational and security evidence within a unified decision process. Second, it clarifies the respective roles of AI-assisted recommendations, deterministic policy controls, automated actions, and human oversight. Third, it grounds the framework in established reliability, resilience, software-delivery, network-analysis, and cybersecurity concepts. Fourth, it demonstrates and evaluates the framework through qualitative scenarios and ex ante design criteria.

2. Related Literature and Research Gap

2.1 Cloud-Native Digital Infrastructure and Resilience

Digital infrastructure comprises computing, software, communications, networks and data resources that are used to support the essential organizational and public functions (Purnamasari et al., 2025). Its resilience relates to its ability to overcome or recover from technical failures, cyber-attacks, demand spikes, configuration mistakes, or supply chain disruptions to maintain or recover to acceptable service levels (Singh, 2025). In cloud-native systems, resilience depends on both architecture and operation: replication, failover, observability, dependency management, fault containment, and recovery readiness must work together. Microservice architecture improves modularity and independent deployment but creates extensive service relationships (Velepucha & Flores, 2023). A failure in a shared database, identity provider, gateway, or messaging service can influence many upstream components. Research on cloud-native resilience therefore emphasizes circuit breakers, bulkheads, retries with backoff, health checks, graceful degradation, canary releases, and automated rollback (Kishore Vadapalli, 2026). These mechanisms reduce local impact only when they are configured with knowledge of service objectives and dependencies.

Latency, traffic, errors and saturation are typical observability targets, along with logs and distributed traces. Service-level indicators translate the measurements chosen to user-oriented measures, and service-level objectives are levels that are acceptable. Error budgets can provide an explicit allowance for unreliability and relate it to release decisions (Hari Dasari, 2025). Resilience also is a time domain. The resilience-triangle tradition views disruption as a manifestation of a system's loss of functionality, and recovery as a return of functionality over time (Gharibzahedi et al., 2024). While two incidents may have the same "peak severity" rating, the consequences may be very different if it's contained and recovered rapidly vs if it continues or spreads. However, cloud-native resilience needs to take into consideration correlated and common-cause failures (Liang et al., 2026). Replicas placed within the same region, dependent on the same identity provider, or built from the same defective image are not truly independent. Availability calculations based on independent components can therefore overestimate resilience when shared dependencies are ignored. There is also a distinction between symptom-based and cause-based evidence (Aldecoa et al., 2023). Traces, logs, deployment events and topology provide an explanation for SLO violations, which represent symptoms visible to users. Both forms are needed: monitoring for causes without considering the user may result in too many alerts, monitoring for symptoms without the cause may result in delayed diagnosis and targeted recovery.

2.2 AI-Assisted Cloud Operations

AIOps uses machine learning, statistical modelling, graph analysis, causal reasoning and automation for operational management (Somayajula, 2025). It has four main purposes: anomaly detection, failure prediction, alert correlation, and root cause analysis as well as capacity forecasting and response recommendation. Thresholds are still useful when there are definite service limits, but fixed thresholds can result in nuisance alarms during normal periods of higher demand or lack of alarms when multiple signals are degraded. AI-assisted methods can analyze patterns across latency, errors, resource saturation, topology, and deployment changes (Song, 2025). Graph-based approaches are especially relevant because microservices form dependency networks rather than independent observations (Tiwari, 2024).

Special care needs to be taken with root-cause analysis. A method can identify the service that is showing the most anomalous behavior without determining what part of the system started the incident. Often the most visible symptoms appear in upstream services, even if the cause is a downstream database, queue or external API. Correlation with a recent deployment may be informative but does not by itself establish causality (Rotari & Kulahci, 2024). AIOps also bring lifecycle obligations. Models should be controlled, monitored, have retraining criteria, and have documented limits and procedures for rollback. A model trained on a topology or traffic pattern may be unreliable following a change in topology (Kim et al., 2024). The governance process should thus document the version of the model that was used for each recommendation and retain fallback procedures if the analytical service cannot be used. Large language models may support log summarization, incident communication, and retrieval of runbook (Ma et al., 2026). Their work can be used for support and may include explanations that aren't always supported or unsafe instructions. Access to tools must be limited, actions created must go through policy checks, and key outputs must be auditable.

2.3 Secure Software Delivery and Supply-Chain Security

Secure Software Delivery is related to the integrity and security of software systems from their inception to build, package, transport, deploy and use (Otieno et al., 2023). DevSecOps is about embedding security activities during the lifecycle, not the last gate in the release cycle (Vidyasagar Vangala, 2025). These are relevant controls, such as source protection, code analysis, dependency scanning, secrets detection, container-image assessment, infrastructure-as-code review, access control, test evidence and runtime monitoring. SBOMs are used to help with component transparency and vulnerability response (Xia et al., 2023). The signatures of artifacts establish integrity and origin, and the attestations and provenance of artifacts provide evidence about how the artifact was built, the inputs, and the builder responsible. SLSA takes this increasing supply-chain assurance as a matter of defined levels and tracks, and the NIST Secure Software Development Framework organizes secure software development practices, which can be combined with various software development lifecycle models (Ghanshyambhai Patel, 2025). CVSS gives a standardized description of the severity of the vulnerability and its environment (Ou & Singhal, 2011). The EPSS can be used to support information about the likelihood of exploitation and known exploitation can be considered via the CISA Known Exploited Vulnerabilities catalogue (Jacobs et al., 2021).

Another issue is the rollback safety. Undoing the operation and going back to the previous version can cause availability to return, but add back a known vulnerability, expired credential, incompatible database state or unsupported dependency (Chu et al., 2025). Recovery choices should thus be compared to the target version and not based on the assumption that an earlier version is necessarily safer. Secure delivery should keep a record of deployable and recoverable artifacts and their current vulnerability and provenance (Hassanshahi et al., 2023). The same goes post disclosure of vulnerabilities. An artifact that was once found acceptable may no longer be acceptable if new evidence is found (Massacci et al., 2022). This means that assurance is continuous and event-driven and not a single gate at the end of the pipeline.

2.4 Policy-Aware Automation and AI Governance

Policy-aware automation can separate out those conditions which can be evaluated in a probability sense and those which reflect explicit organizational requirements (Adityamallikarjunkumar, 2024). An AI model could make an educated guess as to the probability that services are degraded, but it should not replace an explicit rule that an artifact has an acceptable signature, a trusted provenance or a trusted repository (Shieh, 2024). Human supervision should be proportional to consequence and reversibility. This can be preauthorized, such as to increase telemetry or scale a noncritical replica, or may go to a human for approval, such as to isolate an essential public service or override a security block. To support this oversight, explainability reveals the evidence, uncertainty, dependency path and expected consequence of a recommendation (Sadovski et al., 2025).

The NIST AI Risk Management Framework offers a governance framework that is relevant, with the functions of Govern, Map, Measure, and Manage (Wendt, 2025). These functions include the definition of roles, context, measurement, risk tolerance and response. It is essential that they be translated into technical evidence flows, decision rights, policy checks, and audit records in a cloud-native framework. Separation of duties is also a part of accountability. The team that creates an AI model might not be the same team that has the responsibility to approve a policy exception, and the platform team that performs a rollback might not be the same team that has responsibility for the business impact of rolling back a service (Kuehnert et al., 2025). An audit record should document: Evidence seen, versions of model and policy, level of confidence for recommendation, action taken, authorization, and result (Usanovich, 2024). These records can be used to inform learning after the incident and to determine if this framework is consistent over time.

2.5 Design Science Research methodology

Design Science Research is a problem-oriented research approach, which emphasizes the design and evaluation of purposeful artifacts (Botha & Taylor, 2022). It is employed in many research on information-systems and information technology when the goal is to not only give an explanation for an understood phenomenon but to evoke a structured response to an identified problem. Core of Design Science Research is that knowledge can be built up by designing an artifact and by systematically evaluating the usefulness and the inner logic of an artifact (Weigand & Johannesson, 2023). The artifact needs therefore to be associated with a problem and have a reference body of knowledge. Design Science Research thus offers a systematic basis to design technologically oriented artifacts, and to demarcate a proposed design from an operationally validated system (Botha & Taylor, 2022). It places relevance and rigor at the heart of its activities, as well as clarity of artifact creation and demonstration and evaluation, thereby facilitating the creation of structured and defensible conceptual contributions.

2.6 Literature Synthesis and Research Gap

The literature can be organized into four streams: cloud-native resilience, AI-assisted operations, secure software delivery, and AI governance. Each offers mature concepts, but their integration remains limited. Cloud-native resilience provides service-level objectives, observability, fault containment, and recovery mechanisms. AIOps contribute to prediction and diagnosis but introduces uncertainty and model-governance requirements. Secure delivery provides artifact assurance, vulnerability management, and provenance but often stops at deployment. AI governance clarifies oversight and accountability but does not specify how telemetry and software-supply-chain evidence should be combined in a cloud-native control loop.

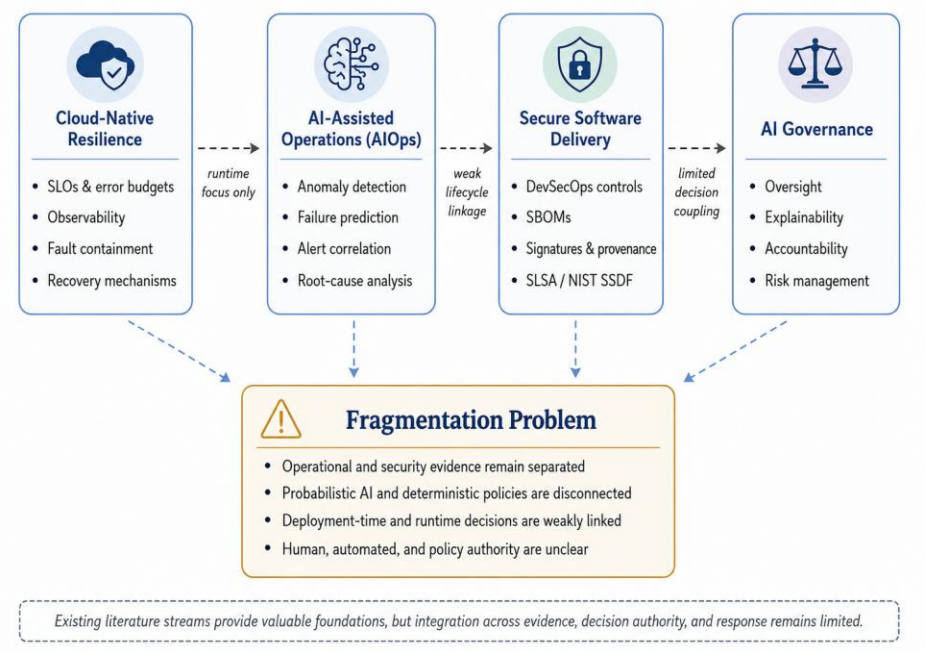

Figure 1. Fragmentation among cloud-native resilience, AIOps, secure delivery, and AI governance

The principal research gap is therefore architectural and procedural. Existing work does not consistently define how service objectives, runtime telemetry, dependency structure, AI analysis, artifact assurance, mandatory security rules, automated actions, and human approval should interact. AICR-SDF addresses this gap by integrating these elements while preserving their different roles and authority.

3. Research Methodology

3.1 Research Design

In this study, Design Science Research Methodology is used in developing a conceptual artifact. Design science is suitable since the research goal is to develop and validate an artifact which can solve a practical problem that is theoretically underpinned. The constructs, layers, relationships, decision roles, proposed measures and implementation propositions are part of the proposed artifact. An integrative literature review is used to build the knowledge base for the development of the artifact. Cloud-native resilience and observability, AIOps and root cause analysis, DevSecOps and secure software delivery, software-supply-chain assurance, AI governance and design-science evaluation are all included. According to relevance, credibility and conceptual contribution, academic studies and authoritative technical sources are considered. Evidence is drawn on the problem being addressed, the technical mechanism, the inputs to the decision, the outputs of the decision, the limitations, the governance implications and relationship to the proposed framework. The results are consolidated and presented as design requirements, as opposed to stand-alone technologies.

Table 1. Application of DSRM activities in the present study

DSRM activity	Application	Output
Problem identification	Fragmentation among resilience, AIOps, secure delivery, and governance	Research problem
Solution objectives	Translation of limitations into design requirements	Requirements and principles
Design and develop	Construction of layers, flows, roles, and theoretical measures	AICR-SDF artifact
Demonstration	Qualitative application to four scenarios	Decision-path walkthroughs
Evaluation	Ex ante assessment using traceability, consistency, alignment, feasibility, and governance	Evaluation matrix
Communication	Journal manuscript and implementation agenda	Research contribution

The review intentionally focuses on current measures and known frameworks. Reliability is based on availability, MT and service level measures. DORA metrics are related to delivery performance. CVSS, EPSS, CISA KEV, SLSA and NIST SSDF are related to

Vulnerability and supply-chain assessment. AI governance is in line with NIST AI RMF. The synthesis process is of a problem-mechanism-requirement type. A recurring problem is identified throughout literature, for example, the evidence is incomplete or automated diagnosis is uncertain. The current mechanisms are then discussed in terms of how much they resolve and how they fall short. Key design requirements state what the proposed artifact will deliver, but without specifying a software product. This process ensures traceability from evidence to design and minimizes the chances of choosing framework layers to complete the need to see a full picture.

3.2 Application of Design Science Research Methodology

The development of AICR-SDF follows the six activities of Design Science Research Methodology shown in Figure 2. The process is supported by an integrative literature review and established reliability, software-delivery, cybersecurity, resilience, and governance standards. Although the activities are presented sequentially, the development process is iterative, allowing findings from demonstration and evaluation to refine the framework architecture and design requirements. The first activity, problem identification and motivation, defines the fragmentation among cloud-native resilience, AIOps, secure software delivery, software-supply-chain assurance, and AI governance. These areas provide useful capabilities independently, but existing approaches offer limited guidance on how their evidence and controls should be integrated within a single deployment and recovery decision process. The output of this activity is the research problem addressed by the study. The second activity, definition of solution objectives, translates the identified problem into requirements for the proposed artifact. These objectives define the capabilities and boundaries that the framework must satisfy. The third activity, design and development, constructs AICR-SDF as the principal research artifact. The identified requirements are organized into six interconnected layers covering critical digital services, cloud-native infrastructure, observability and evidence, AI resilience intelligence, secure software delivery, and policy, response, and governance. The fourth activity, demonstration, applies the framework to four qualitative scenarios: a normal verified deployment, gradual service degradation, an unverified software artifact, and cascading microservice failure. The scenarios illustrate how different combinations of operational conditions, security evidence, service dependencies, and mandatory policies activate different framework pathways and responses.

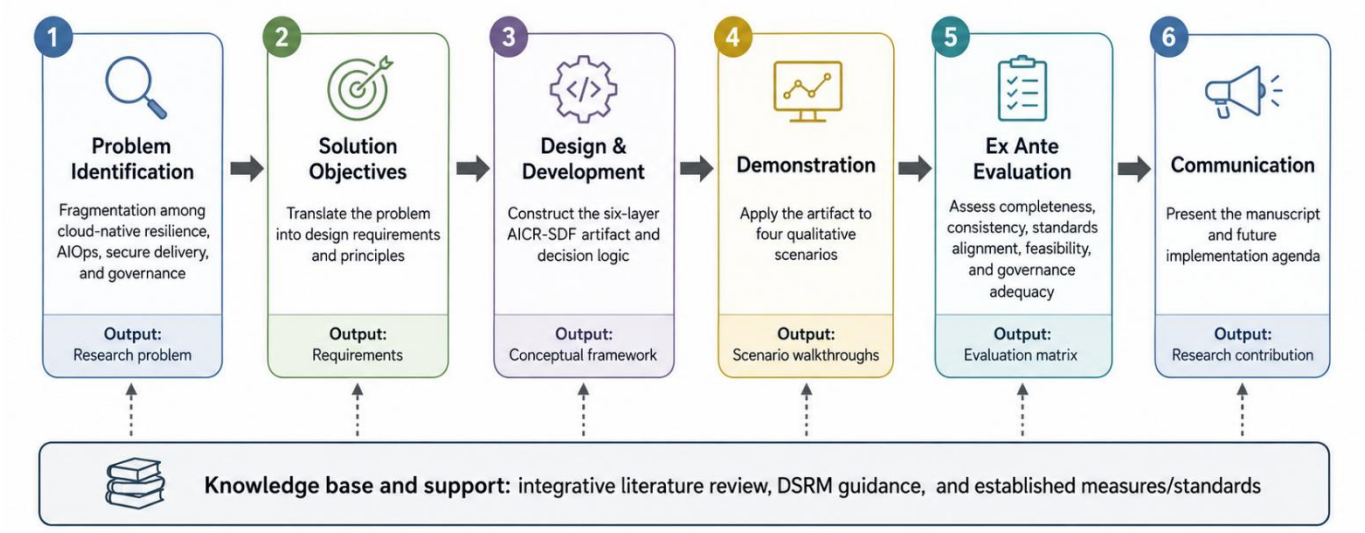

Figure 2. Research methodology and adapted DSRM process

The fifth activity, evaluation, examines the conceptual artifact before implementation. The evaluation focuses on whether the framework covers its design requirements, maintains logical consistency, supports the selected scenarios, aligns with established standards, provides adequate governance controls, and can be implemented using existing technology categories. The sixth activity, communication, presents the research problem, theoretical foundations, design process, framework architecture, demonstration, evaluation, limitations, and future validation requirements through the manuscript. The outcome is a documented conceptual contribution and a foundation for subsequent prototyping and empirical assessment.

The evaluation strategy is informed by design-science evaluation principles and is formative, ex ante, and predominantly analytical. The evaluand is the AICR-SDF conceptual architecture, including its six layers, information flows, decision roles, and governance mechanisms. The evaluation does not attempt to measure operational effectiveness because the framework has not yet been implemented.

4. Development of the Proposed Conceptual Framework

4.1 Design Requirements and Principles

Literature synthesis produced nine design requirements for AICR-SDF. Collectively, these requirements address four core needs: integration of operational and software-assurance evidence, awareness of service dependencies, governed use of AI-assisted analysis, and accountable execution of resilience actions. Table 2 summarizes the purpose of each requirement and identifies the framework components through which it is addressed.

Table 2. Framework design requirements and corresponding components

Requirement	Purpose	Framework support
Integrated evidence	Combine runtime and software-assurance information	Observability and Evidence; Secure Software Delivery
Dependency awareness	Interpret local failure within service topology	Cloud-Native Infrastructure; AI Resilience Intelligence
Predictive intelligence	Identify emerging degradation and likely causes	AI Resilience Intelligence
Artifact assurance	Verify integrity, provenance, components, and build evidence	Secure Software Delivery
Policy enforcement	Protect non-negotiable requirements	Policy, Response, and Governance
Explainability	Support review of AI recommendations	AI Resilience Intelligence; Governance
Human oversight	Control high-impact and exceptional actions	Policy, Response, and Governance
Recovery support	Connect evidence with executable response	Infrastructure; Governance
Auditability	Preserve evidence, decisions, actions, and outcomes	Observability and Evidence; Governance

The requirements are implemented through five design principles. Evidence integration ensures that deployment and recovery decisions are informed by both operational and security information. Separation of authority distinguishes AI-generated recommendations from deterministic controls, automated actions, and accountable human approval. Layered responsibility assigns clear functions to each part of the framework while preserving information exchange across layers. Risk-proportionate automation allows reversible and low-impact actions to be automated while reserving high-impact or uncertain decisions for human review. Continuous feedback returns action outcomes to observability, policy review, audit processes, and future model refinement.

4.2 Framework Architecture and Layers

AICR-SDF is comprised of six interrelated layers. The Critical Digital Services Layer specifies service objectives, users, criticality, acceptable disruption, regulatory requirements, and authority to make decisions. It makes sure that a technical signal is translated in accordance with the mission supported by the service. The Cloud-Native Infrastructure Layer includes microservices, containers, orchestration platforms, gateways, service meshes, databases, messaging systems, networks, and external resources. It gives topology, health and status and performs approved actions like scaling, traffic redirection, isolation, restart and rollback. The Observability and Evidence Layer aggregates metrics, logs, traces, events, deployment metadata, dependency maps, test results, vulnerabilities found, signatures, attestations, provenance, policy outcomes, and decision records. It is designed to link evidence over time, software versions and service relationships.

AI Resilience Intelligence Layer provides anomaly detection, failure prediction, event correlation, dependency-aware analysis, and root cause ranking. Secure Software Delivery Layer assesses source and build controls, testing, component inventories, vulnerability status, artifact signatures, attestations, provenance, and delivery-policy compliance. It separates the advisory risk information from mandatory pass-or-fail controls. The Policy, Response and Governance Layer integrate service criticality, AI recommendations, security benchmarks, rules, action permissions and human authorization. It chooses or approves promotion, monitoring, scaling, isolation, rollback, quarantine or escalation.

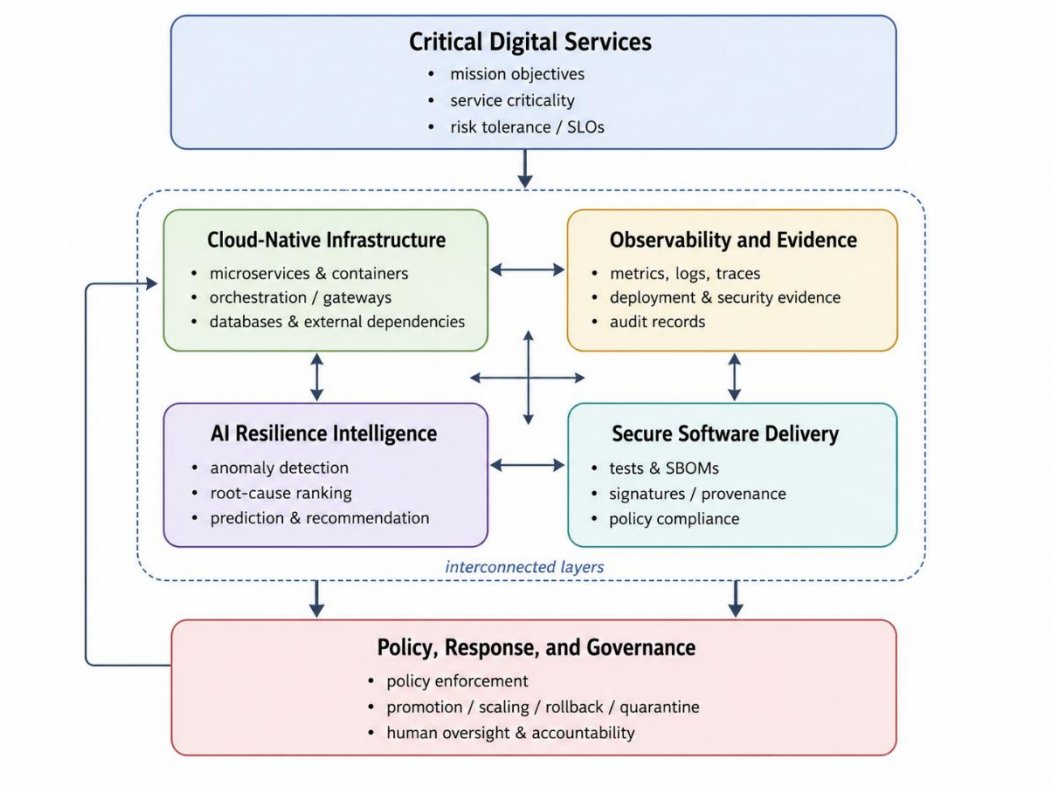

Figure 3. Proposed AI-powered cloud-native resilience and secure-delivery conceptual framework

The six-layer design should not be interpreted as a strictly linear pipeline. Operational evidence may flow directly from infrastructure to observability, while a mandatory signature failure can move from the secure-delivery layer directly to policy enforcement. The AI layer may request additional traces, and the governance layer may reduce rollout scope before a final decision. The architecture is therefore layered for responsibility but event-driven in operation. The framework is also technology-agnostic at the conceptual level. A future implementation may use different telemetry platforms, policy engines, graph stores, vulnerability services, or machine-learning models. Interoperability depends on common identities for services, versions, artifacts, deployments, policies, and incidents. These identities allow evidence generated at different lifecycle stages to be correlated. An implementation should preserve the distinction between raw evidence, derived indicators, model output, and final decisions. This separation improves explainability and prevents a model-generated label from being mistaken for an observed fact.

4.3 Information and Decision Flows

There is a continuous control loop of information. Service objectives and risk tolerance are defined in the Critical Digital Services Layer. Runtime conditions and topology are generated by the infrastructure. Build, testing, vulnerability, and provenance evidence is created during delivery pipelines. The Operational Context Sources expose and correlate these sources and pass the operational context to the AI layer and the policy relevant evidence to the governance layer. The AI layer extracts the uncertainty from each situation and generates a recommendation with high levels of confidence and explanation. Secure delivery layer provides assurance status and failure detection of required policies. The governance layer thinks of way through either of the paths. A probabilistic recommendation can be used to influence monitoring, scaling or rollback, however a deterministic rule can prevent an artifact from being disabled, regardless of the operational stability prediction.

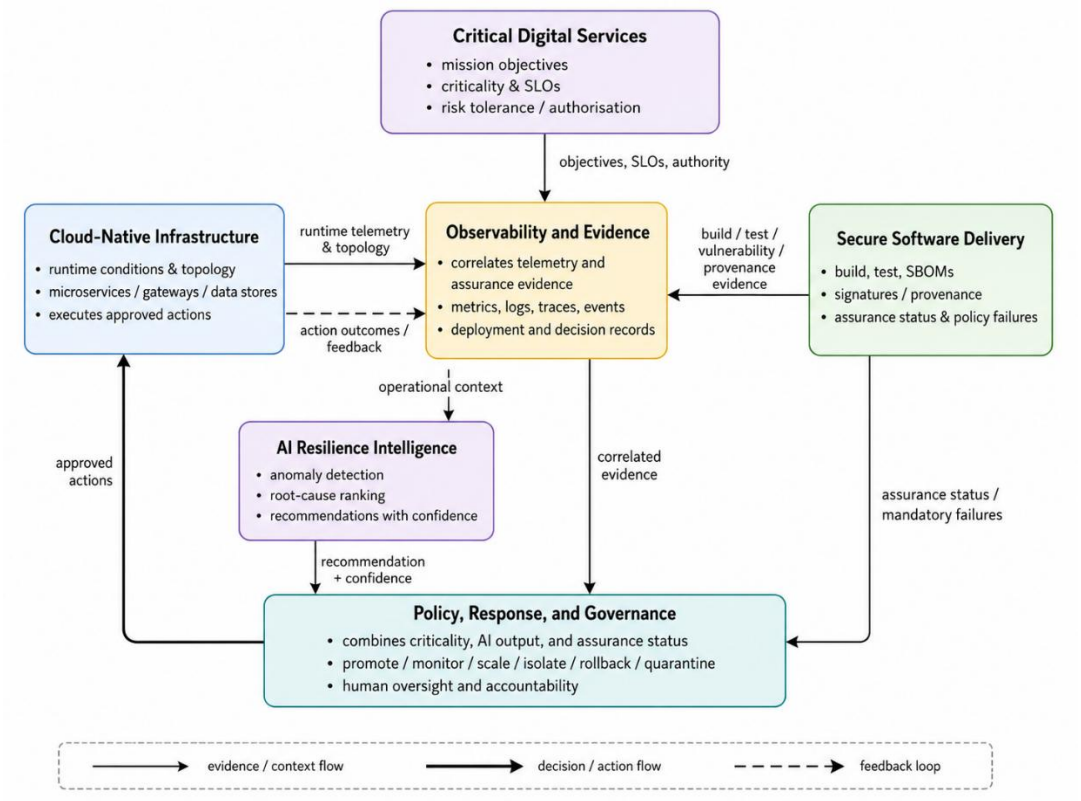

Figure 4. Evidence and decision flows across the framework

The infrastructure or delivery pipeline runs the actions that have been approved. Their results go back to the evidence layer. This feedback is used to help review incidents, manage SLOs, revise policies, and develop future models in an empirical manner. The decision loop can be applied on different time scales. Pre-deployment controls include source, build, test, vulnerability, signature, and provenance evidence. Canary and control behavior is compared during deployment time and might limit the scope for deployment. Runtime controls are used to re-evaluate SLOs, error-budget burn, service topology and incident evidence. Post-incident controls support/preserves decision trail and updates runbooks/policies/priorities to model development. One of the main goals of the evidence layer is continuity across these time scales. The framework involves identity correlation also. The attribute values of service names, artifact digests, deployment identifiers, policy versions, and incident records must all align with an identical release and operational context. Vulnerability finding cannot be reliably mapped back to the running instance without this linkage and a rollback decision cannot be made for whether the targeted artifact meets current security policy terms.

4.4 AI, Policy, Automation, and Human Decision Roles

There are four roles in the framework. AI recommends actions that tackle uncertain patterns and prioritize available actions. Explicit control mechanisms like validity of the signature or availability of the provenance or repository authorization are determined by deterministic controls. Automatically Executable Actions have limited effect, reversible and pre-authorized. Human-authorized decisions include high consequence decisions, or policy exceptions, or decisions where the evidence is uncertain, or decisions that have an impact on critical services. This separation is important because a numerical risk estimate and a security policy do not represent the same form of knowledge. A model predicts; a policy constrains; an automation mechanism executes; and an accountable actor authorizes exceptional or high-impact decisions.

5. Theoretical Measures and Policy-Aware Decision Mechanism

The framework does not introduce an unvalidated composite risk score. Instead, it uses established theoretical measures and benchmarks to define what a future implementation should observe and how evidence can be interpreted. These measures are not combined into one universal number because they represent different constructs and have different assumptions.

5.1 Reliability, Availability, and Service-Level Measures

Availability and reliability provide the core operational foundation. Availability can be represented as:

$$A = \frac{MTBF}{MTBF + MTTR}$$

where MTBF is mean time between failures and MTTR is mean time to repair or restore(Yuan & Lu, 2022). The equation shows that availability can improve by increasing the interval between failures or reducing recovery time. In practice, incident definitions and measurement windows must be specified clearly. Under a constant failure-rate assumption, reliability over time can be represented as:

$$R(t) = e^{-\lambda t}$$

where λ is the failure rate and t is the operating interval.(Uy, 2016) The exponential model is a theoretical baseline and should not be applied when failure rates vary substantially over time. For a serial service path with independent component availability, system availability can be approximated as:

$$A\_system = A1 \times A2 \times \ldots \times An$$

This relationship illustrates why long dependency chains can reduce end-to-end availability (Qiu, 2019). Independence is a strong assumption; shared infrastructure and correlated failures require more advanced modelling. Service-level objectives translate reliability into user-oriented targets. When the SLO is expressed as a proportion, the error budget is (Kumar Panchalingala, 2025):

$$Error\ Budget = 1 - SLO$$

Burn rate indicates how quickly the service consumes this budget:

$$Burn\ Rate = \frac{Observed\_Error\_Rate}{Allowed\_Error\_Rate}$$

A burn rate greater than one indicates that the service is consuming the error budget faster than permitted for the measurement window (Mata, 2018). Multi-window burn-rate analysis provides a defensible basis for monitoring and escalation without inventing a new threshold system.

5.2 Resilience and Software-Delivery Benchmarks

The resilience-triangle perspective represents the loss and recovery of service functionality over time. Let Q(t) represent normalized service performance, where one represents the required performance level. Resilience loss over a disruption interval can be expressed as:

$$Resilience\ Loss = \int_{t0}^{t1} [1 - Q(t)]\, dt$$

The measure captures both the depth of performance loss and the duration of recovery (Yeligeti et al., 2025). For cloud-native systems, Q(t) may be represented by a service-level indicator such as successful request proportion, transaction completion, or a composite service objective established by the organization. The formulation does not predict failure; it provides a theoretically grounded way to compare incident impact after empirical data become available. DORA software delivery metrics provide established benchmarks for delivery performance and stability. Relevant measures include deployment frequency, change lead time, change fail percentage, and failed deployment recovery time. These metrics help evaluate whether secure-delivery controls improve assurance without producing unacceptable delay or recovery burden. Change fail percentage can be represented as:

$$Change\ Fail\ Percentage = \frac{Failed\ Production\ Changes}{Total\ Production\ Changes}$$

The framework should use these metrics as future evaluation criteria. DORA measures also expose possible trade-offs. Stronger supply-chain controls should not be assumed to improve resilience if they create excessive lead time, unreliable emergency changes, or recovery delay. Conversely, high deployment frequency should not be treated as success when change fail percentage is unacceptable. Future evaluation should therefore examine delivery throughput and stability together and compare results before and after adoption framework.

5.3 Dependency Criticality

Microservice dependencies can be represented as a directed graph in which services are nodes and calls or data dependencies are edges. Established centrality measures can identify services whose failure may have disproportionate effects. Normalized degree centrality is:

$$C_D(v) = degree(v) / (n - 1)$$

where degree(v) is the number of relevant connections of service v and n is the number of services. In directed graphs, in-degree and out-degree can be examined separately. Betweenness centrality is:

$$C_B(v) = sum\ over\ s\ and\ t\ of\ [sigma_st(v) / sigma_st]$$

where sigma_st is the number of shortest paths between services s and t, and sigma_st(v) is the number of those paths passing through v. A service with high betweenness may act as a bridge between otherwise separate service groups and therefore deserves stronger monitoring, redundancy, or isolation planning. Centrality does not itself measure failure probability. It represents structural importance and should be interpreted with runtime telemetry, criticality, and architecture knowledge.

5.4 Cybersecurity, Supply-Chain, and AI-Governance Benchmarks

Cybersecurity risk is treated as a function of likelihood and impact, consistent with established risk-assessment practice:

$$Risk = f\ (Likelihood, Impact)$$

A simple risk matrix may approximate this relationship as Likelihood × Impact, but the framework does not claim that such multiplication captures all technical and organizational context. Vulnerability evidence should use recognized benchmarks. CVSS describes vulnerability severity and allows threat and environmental context. EPSS estimates the probability of exploitation within a defined period, while the CISA Known Exploited Vulnerabilities catalogue identifies vulnerabilities with evidence of active exploitation. These sources should inform prioritization but remain distinct. Software supply chain assurance is represented through NIST, SSDF practices and SLSA levels or tracks. Artifact signatures, SBOMs, attestations, and provenance provide evidence that can be evaluated by deterministic policies. For example, an organization may require verified provenance and an approved builder for production deployment. Such a rule should be represented as a pass-or-fail condition rather than converted into an arbitrary percentage.

5.5 Policy-Aware Response Mechanism

The policy-aware response mechanism leverages implemented measures, rather than giving them a sum at the end of the day that the computer didn't understand. Promotion must have successful mandatory assurance checks, acceptable SLO and error budget, and not show any negative impact of staged deployment degrading the service. Use of monitoring is suitable if evidence is incomplete, burning rate is rising but not reaching a critical level, or model confidence is not high. Scaling is suitable if traffic and Saturation suggest pressure on capacity and no software / dependency fault is found. Isolation is used for a component that is also structurally critical, fails to function properly and can be isolated without even more significant damage to the mission. Rollback can be used whenever a change has been made recently and results in degradation, but a previous version is known to exist. This is because, if an artifact does not pass a non-negotiable integrity or provenance rule, it is put on quarantine, which is a requirement. Human Review is required if evidence is inconsistent, or if services are critical, a vulnerable version may be re-redeployed in the case of rollback or there is a requested Policy Exception. Thus, the framework adopts existing measures for reasoning support, and policies and responsible/answerable roles are in place to ensure authority.

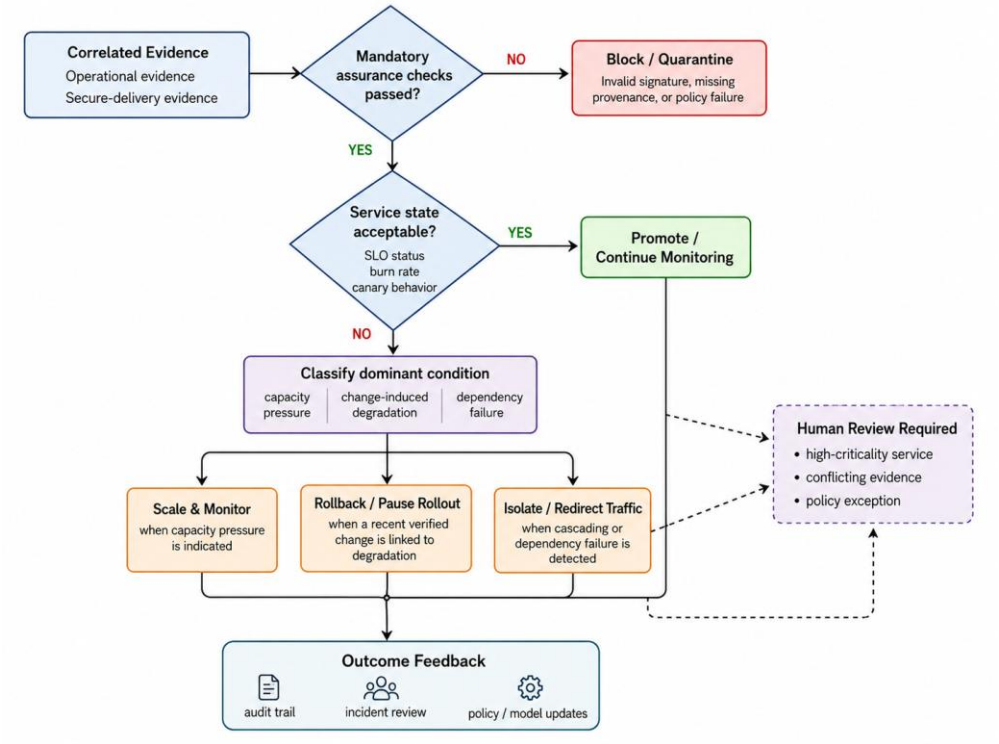

Figure 5. Policy-aware deployment and recovery decision process

It is advisable to record response rules as organizational policies, not as universal 'constants'. An error-budget burn condition that stops the rollout of a low-risk feature may not be appropriate for stopping an emergency service. Arguably, exposures, evidence of exploits, compensating controls and mission impacts can vary an action required based on mitigation requirement, which is the CVSS score to this point. The framework offers a structure for how to combine this evidence, but it does not dictate what the tolerances and/or authorizations of the adopting organization will be. Staged response should also be used in future implementation. Monitoring is one way to boost evidence gathering prior to disruptive action. There are "boundary costs" and "saturation" when it comes to scaling. Isolation can be applied to a feature or dependency, but not the entire service. You can roll back on canary/release region level. Staged action minimizes the risk of an inappropriate recommendation leading to a larger incident.

6. Ex Ante Evaluation and Demonstration with Scenario approach

6.1 Ex Ante Framework Evaluation

The ex ante evaluation is based on a formative design-science logic. Requirements coverage reviews and answers the question of do all of the requirements from the literature relate to a layer, flow, policy or role. Internal consistency explores contradictory authority or function duplication in the framework. In the process of considering alignment of standards, attention is paid to whether the framework includes clearly established measures and practices or unidentified constructs? Scenario applicability examines whether different conditions activate different evidence and responses. Technical implementability examines whether existing observability, orchestration, security, policy, and AI tools could realize the proposed functions. Governance adequacy examines explainability, auditability, human roles, and policy precedence. The evaluation supports the plausibility and traceability of the design. It cannot establish predictive accuracy, operational utility, user acceptance, cost, computational overhead, or sector-wide generalizability. These properties require prototype implementation, expert review, controlled fault injection, security testing, and field evaluation.

Table 3. Ex ante evaluation matrix

Evaluation property	Question	Evidence in the conceptual artifact
Requirements coverage	Does every literature-derived requirement have a design response?	Requirement-to-layer mapping
Internal consistency	Are authority, evidence flow, and response logic non-contradictory?	Separation of AI, policy, automation, and human roles
Standards alignment	Are measures and controls recognised and defensible?	SLOs, DORA, CVSS, SLSA, SSDF, AI RMF, centrality, resilience loss
Scenario applicability	Do distinct situations activate different decision paths?	Four qualitative walkthroughs
Technical implementability	Can available technology categories realise the functions?	Observability, orchestration, policy engines, supply-chain tools, AI services
Governance adequacy	Are explanation, audit, oversight, and policy precedence represented?	Governance layer and decision-role model

An evaluation program should be carried out in stages to develop in the future. The completeness and policy realism can be measured through expert review. Evidence integration and decision latency are amenable to laboratory tests on prototype. Controlled experiments can use fault injections, vulnerable components, invalid provenance, and rollback conditions. Naturalistic evaluation can then examine operator trust, false alarms, incident outcomes, and organizational adoption. This staged approach is consistent with the transition from ex ante artificial evaluation toward ex post naturalistic evaluation in design-science research.

6.2 Qualitative Demonstration Scenarios

The logic underlying the framework is illustrated via four qualitative scenarios. These are structured walkthroughs that demonstrate which evidence sources, benchmarks, policies and decision roles are engaged. A normal verified deployment is denoted as scenario 1. The artifact is from an approved source, signed, requires provenance, tests are completed, and it is acceptable in terms of vulnerability. All Canary service level indicators are still within SLO limits, no dependency anomaly has been detected, and error-budget consumption is normal. The framework allows for promotion and ongoing monitoring. Scenario 2 is the step down of service. Since deployment, latency and saturation increases and security and provenance checks are still valid. The evidence of resource and tracing suggests pressures of capacity and burn-rate monitoring suggests accelerated error-budget consumption it is most likely a pressure of capacity. The framework suggests for controlled scaling and improvement of monitoring. If the burn rate continues to be high OR the Canary is moving off of the control, the rollout is halted and a rollback is evaluated.

Table 4. Qualitative scenario conditions and framework responses

Scenario	Evidence condition	Applicable established measure or policy	Framework response
Normal verified deployment	Valid provenance and signature; canary within SLO; normal error-budget use	SLO, error budget, canary comparison, SSDF/SLSA policy	Promote and monitor
Gradual service degradation	Rising latency and saturation; accelerated burn rate; assurance checks valid	SLIs, burn rate, traces, DORA recovery measure	Scale and monitor; pause or rollback if degradation persists
Unverified artifact	Runtime appears stable but provenance or signature requirement fails	Deterministic artifact-assurance policy	Block or quarantine
Cascading microservice failure	Timeout propagation through a high-centrality dependency	Tracing, burn rate, degree/betweenness centrality	Isolate, redirect, and roll back with human approval when critical

An unverified Software artifact is represented by scenario 3. Telemetry data was obtained at runtime but cannot be shown because its provenance was not supplied or is not verified by the signature. The condition is a violation of a deterministic

deployment policy, so the artifact is blocked or quarantined. The stability in operation is not more important than assurance. Scenario 4 is microservice's chain reaction of failure. A downstream service is degraded due to timeouts, retries and increased error-budget in dependent services. Traces and graph structure determine that there is a high-centrality component within the failure path. Circuit Breaking or isolation, traffic Redirection if available and Rollback if the incident is related to a recent documented change are the recommended actions of the framework. If isolating impacts an essential service, there is need for human authorization.

7. Discussion

AICR-SDF addresses a persistent fragmentation in cloud-native operations by integrating areas that are commonly managed through separate technical platforms, organizational teams, and governance processes. Cloud reliability teams typically focus on observability, service-level objectives, incident response, and recovery, while software-security teams focus on vulnerabilities, provenance, build integrity, and delivery controls. AI-assisted operational tools add anomaly detection, prediction, and root-cause analysis, whereas governance functions define policy, authority, accountability, and acceptable risk. The principal contribution of the proposed framework is not the introduction of these capabilities individually, but the specification of how they should exchange evidence and how their respective decision roles should differ. This integration is important because operational resilience and software assurance can produce conflicting signals. A deployment may satisfy latency and availability objectives while failing a mandatory artifact-signature or provenance requirement. Conversely, a verified artifact may introduce a memory leak, dependency bottleneck, or cascading service failure after deployment. Treating either operational evidence or software-assurance evidence as sufficient can therefore produce incomplete decisions. AICR-SDF addresses this problem by maintaining separate evidence paths while bringing them together within a common governance process. This allows probabilistic operational assessments to inform decisions without weakening deterministic security requirements.

The framework's theoretical foundation also strengthens its conceptual validity. Rather than introducing an unsupported universal score, AICR-SDF draws on established measures that represent distinct constructs. Availability and reliability functions describe continuity of service. Service-level objectives, error budgets, and burn rate indicate whether reliability consumption remains within acceptable limits. Resilience-loss measures represent the depth and duration of degraded performance. DORA metrics capture software-delivery capability and recovery performance. Graph centrality supports analysis of service dependency and propagation potential. Likelihood–impact assessment provides a recognized basis for contextual risk reasoning, while CVSS, SLSA, SSDF, and the AI Risk Management Framework provide established foundations for vulnerability assessment, supply-chain assurance, secure development, and AI governance. The framework does not assume that these measures should be reduced to one composite index. Their meanings differ and combining them without empirical justification could obscure rather than improve decision quality. A high burn rate indicates that an error budget is being consumed rapidly, but it does not prove the root cause. A service with high centrality may require stronger redundancy and containment controls even when it has not failed. A critical vulnerability score may influence prioritization, but the deployment decision may also depend on exploitability, exposure, reachability, service criticality, and available mitigations. The framework therefore treats these measures as complementary evidence interpreted through explicit decision rules.

This distinction has practical importance for critical digital services. In high-consequence environments, policy precedence must remain visible. A low observed error rate cannot compensate for an unverified artifact, while a high availability target cannot justify bypassing required provenance controls. Similarly, automated recovery should not be applied uniformly. Scaling may be appropriate when workload pressure is the dominant problem, but it may worsen a dependency failure or reproduce a defective release. Isolation may contain propagation risk, yet it may also interrupt a critical function. AICR-SDF therefore links technical evidence with service criticality, action reversibility, and decision authority. The framework also adopts a deliberately bounded role for artificial intelligence. AI contributes where uncertainty and multidimensional evidence make manual interpretation difficult. It can support anomaly detection, failure prediction, probable-cause ranking, dependency-aware diagnosis, and response recommendation. However, AI output remains contingent on telemetry quality, model design, environmental stability, and the availability of representative failure data. Rare failures, architecture changes, and incomplete observability can reduce confidence. For this reason, explanation, confidence reporting, escalation, and policy constraints are treated as required design properties rather than optional governance additions.

From an organizational perspective, the framework may provide a shared decision model for platform engineering, software development, cybersecurity, risk management, and service ownership. Its layered structure clarifies which evidence is produced, where it is interpreted, which controls are mandatory, and which actions require approval. This can reduce ambiguity during deployment and incident response, particularly when operational and security priorities conflict. It may also improve auditability by linking service identifiers, artifact digests, deployment records, policy versions, AI recommendations, approvals, and response outcomes within the same operational context. The research nevertheless has several limitations. AICR-SDF remains a conceptual

artifact and has not been implemented or tested with production telemetry. Literature synthesis is integrative rather than a formal systematic review or meta-analysis. The selected standards and measures also require adaptation to organizational context. Service graphs evolve continuously, service-level objectives must reflect user outcomes, and software-security policies may conflict with availability or emergency-response priorities. The feasibility of cross-layer evidence correlation, model performance, policy execution, and operator acceptance therefore remains to be established empirically.

Future research should operationalize the six layers through a prototype architecture and define common data schemas for telemetry, deployment evidence, policy status, and decision records. Alternative AI techniques should be compared for anomaly detection, failure prediction, and root-cause analysis. Policy conflicts should be tested under controlled deployment and failure conditions, and expert panels should evaluate the adequacy of governance boundaries. Organizational case studies could then examine usability, trust, response time, false-positive cost, recovery effectiveness, and integration overhead. The conceptual contribution is therefore not a claim that one architecture resolves all forms of cloud risk. Its value lies in defining a coherent integration problem, identifying defensible technical anchors, and establishing propositions for future evaluation. The framework proposes that decisions become more context-aware when operational and software-assurance evidence are correlated, that policy precedence prevents probabilistic models from weakening mandatory controls, and that dependency-aware evidence improves prioritization of containment and recovery. These propositions provide a clear basis for prototype development and empirical validation.

8. Conclusion

This research developed the AI-Powered Cloud-Native Resilience and Secure Delivery Framework (AICR-SDF) to integrate cloud-native resilience, AI-assisted operational intelligence, secure software delivery, software supply-chain assurance, and policy-governed response within a single conceptual architecture. The framework organizes these capabilities into six interconnected layers and distinguishes the roles of analytical recommendation, deterministic policy enforcement, automated execution, and accountable human authority. The proposed framework contributes a structured approach for combining operational telemetry, service dependencies, delivery evidence, security controls, and governance requirements in support of deployment and recovery decisions. Its design is grounded in established reliability, resilience, software-delivery, network-analysis, cybersecurity, and AI-governance concepts. The qualitative scenarios and ex ante evaluation indicate that the framework is internally coherent, traceable to literature-derived requirements, aligned with relevant standards, and technically feasible in principle.

The research remains conceptual and does not establish operational effectiveness. Future research should implement the framework as a prototype, define organization-specific service-level objectives and policy rules, integrate recognized software-security evidence, and evaluate its performance through controlled experiments, expert assessment, and representative operational data. Such work would determine the framework's practical value in improving the resilience and secure delivery of cloud-native digital services.

Funding: This research received no external funding
Conflicts of Interest: The authors declare no conflict of interest.
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Adityamallikarjunkumar, A. (2024). Compliance-First Automation in the Public Sector. International Journal of Emerging Research in Engineering and Technology, 5(2). https://doi.org/10.63282/3050-922x.ijeret-v5i2p108

[2] Aldecoa, C., Bettelli, G., Bilotta, F., Sanders, R. D., Aceto, P., Audisio, R., Cherubini, A., Cunningham, C., Dabrowski, W., Forookhi, A., Gitti, N., Immonen, K., Kehlet, H., Koch, S., Kotfis, K., Latronico, N., MacLullich, A. M. J., Mevorach, L., Mueller, A., … Spies, C. D. (2023). Update of the European Society of Anaesthesiology and Intensive Care Medicine evidence-based and consensus-based guideline on postoperative delirium in adult patients. European Journal of Anaesthesiology, 41(2), 81–108. https://doi.org/10.1097/eja.0000000000001876

[3] Botha, L., & Taylor, E. (2022, March 12). USING DESIGN SCIENCE RESEARCH AS PARADIGM FOR INFORMATION SYSTEM RESEARCH: AN APPLICATION. https://doi.org/10.33965/is2022_202201l010

[4] Chu, D., Balasubramanian, A., Bao, D., Crooks, N., Howard, H., Katahanas, L., & Ponnapalli, S. (2025). Rollbaccine : Herd Immunity against Storage Rollback Attacks in TEEs [Technical Report. In ArXiv.org. American Chemical Society. https://doi.org/10.48550/arxiv.2505.04014

[5] Devalla, S. (2025). AI-Driven Telemetry Analytics for Predictive Reliability and Privacy in Enterprise-Scale Cloud Systems. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 6(2). https://doi.org/10.63282/3050-9262.ijaidsml-v6i2p114

[6] Ghanshyambhai Patel, D. (2025). Software Supply Chain Security: Implementing SLSA Compliance in CI/CD Pipelines. International Journal for Research Trends and Innovation, 10(7). https://doi.org/10.56975/ijrti.v10i7.205129
[7] Gharibzahedi, S. M. T., Kamran Jalilpoor, Azad, S., Daneshvar, M., Sepasian, M. S., Mohammadi-Ivatloo, B., & Shafie-khah, M. (2024). Future Modern Distribution Networks Resilience. Elsevier.
[8] Hari Dasari. (2025). SITE RELIABILITY ENGINEERING PRACTICES FOR ERROR BUDGET MANAGEMENT IN LARGE-SCALE SYSTEMS. International Journal of Applied Mathematics, 38(5s), 991–1001. https://doi.org/10.12732/ijam.v38i5s.366
[9] Hassanshahi, B., Mai, T. N., Michael, A., Selwyn-Smith, B., Bates, S., & Krishnan, P. (2023). Macaron: A Logic-based Framework for Software Supply Chain Security Assurance. 29–37. https://doi.org/10.1145/3605770.3625213
[10] Jacobs, J., Romanosky, S., Edwards, B., Adjerid, I., & Roytman, M. (2021). Exploit Prediction Scoring System (EPSS). Digital Threats: Research and Practice, 2(3), 1–17. https://doi.org/10.1145/3436242
[11] Kaur, K., Guillemin, F., & Sailhan, F. (2022). Container placement and migration strategies for cloud, fog, and edge data centers: A survey [Review of Container placement and migration strategies for cloud, fog, and edge data centers: A survey]. International Journal of Network Management. https://doi.org/10.1002/nem.2212
[12] Kim, D.-W., Shin, G.-Y., Jang, Y.-H., Cho, S., Kim, K., Kang, J., & Han, M.-M. (2024). Framework for Network Topology Generation and Traffic Prediction Analytics for Cyber Exercises [Review of Framework for Network Topology Generation and Traffic Prediction Analytics for Cyber Exercises]. IEEE Access, 12, 23869–23880. https://doi.org/10.1109/access.2023.3344170
[13] Kishore Vadapalli, V. L. N. (2026). Designing Cloud-Native Distributed Systems for Zero-Downtime Enterprise Platforms [Review of Designing Cloud-Native Distributed Systems for Zero-Downtime Enterprise Platforms]. International Journal of Emerging Trends in Computer Science and Information Technology, 7, 92–103. https://doi.org/10.63282/3050-9246.ijetcsit-v7i2p113
[14] Kuehnert, B., Kim, R., Forlizzi, J., & Heidari, H. (2025). The ``Who", ``What", and ``How" of Responsible AI Governance: A Systematic Review and Meta-Analysis of (Actor, Stage)-Specific Tools. 2991–3005. https://doi.org/10.1145/3715275.3732191
[15] Kumar Panchalingala, A. (2025). Site Reliability Engineering (SRE): Bridging the Gap Between Development and Operations. Technix International Journal for Engineering Research, 12(5). https://doi.org/10.56975/tijer.v12i5.158197
[16] Li, X., Li, J., Yuan, C., Guo, S., & Wang, Z. (2022). Digital Infrastructure [Review of Digital Infrastructure]. Applied Economics and Policy Studies, 39–55. https://doi.org/10.1007/978-981-16-8527-9_3
[17] Liang, S., Chen, P., Tian, B., Tan, G., Xu, M., Qu, Y., Zhao, Y., Shang, Y., & Tan, C. (2026). MetaRCA: A Generalizable Root Cause Analysis Framework for Cloud-Native Systems Powered by Meta Causal Knowledge. Proceedings of the ACM on Software Engineering, 3(FSE), 138–159. https://doi.org/10.1145/3797069
[18] Ma, Z., Yang, J., & Chen, T.-H. (2026). LLM4Log: A Systematic Review of Large Language Model-based Log Analysis. arXiv.Org. https://arxiv.org/abs/2604.16359
[19] Massacci, F., Nikiforakis, N., Pashchenko, I., Sabetta, A., & Wang, V. (2022). Introduction to the Special Issue on Vulnerabilities. Digital Threats: Research and Practice, 3(4), 1. https://doi.org/10.1145/3580605
[20] Mata, A. J. (2018). Burning Cost. In Wiley StatsRef: Statistics Reference Online (pp. 1–4). Wiley. https://doi.org/10.1002/9781118445112.stat04732.pub2
[21] Otieno, M., Odera, D., & Jairus Ekume Ounza. (2023). Theory and practice in secure software development lifecycle: A comprehensive survey. World Journal Of Advanced Research and Reviews, 18(3), 053–078. https://doi.org/10.30574/wjarr.2023.18.3.0944
[22] Ou, X., & Singhal, A. (2011). The Common Vulnerability Scoring System (CVSS) (pp. 9–12). Springer New York. https://doi.org/10.1007/978-1-4614-1860-3_3
[23] Purnamasari, R., Hasanudin, A. I., Zulfikar, R., & Yazid, H. (2025). Technological infrastructure and financial resource availability in enhancing public services and government performance: The role of digital innovation adoption in Indonesia [Review of Technological infrastructure and financial resource availability in enhancing public services and government performance: The role of digital innovation adoption in Indonesia]. Social Sciences & Humanities Open, 11, 101621. https://doi.org/10.1016/j.ssaho.2025.101621
[24] Qiu, X. (2019). Reliability and Availability Engineering: Modeling, Analysis, and Applications by Kishor S. Trivedi and Andrea Bobbio. 2017 (ISBN 9781107099500). International Journal of Performability Engineering, 15(4). https://doi.org/10.23940/ijpe.19.04.p22.12631264
[25] Rotari, M., & Kulahci, M. (2024). Correlation to causality [Review of Correlation to causality]. Quality Engineering, 1–11. https://doi.org/10.1080/08982112.2024.2372489
[26] Sadovski, E., Aviv, I., & Hadar, I. (2025). Navigating the Human-oversight Dilemma in AI-based Systems. 454–461. https://doi.org/10.1109/rew66121.2025.00069
[27] Sayles, J. (2024). Aligning AI Governance, AI Development Lifecycle, and Systems Development Lifecycle Processes [Review of Aligning AI Governance, AI Development Lifecycle, and Systems Development Lifecycle Processes]. Apress eBooks, 383–408. https://doi.org/10.1007/979-8-8688-0983-5_21
[28] Shieh, W. (2024). Editorial Ensuring Reliability, Security, and Trust for Enterprises. IEEE Transactions on Reliability, 73(2), 805–807. https://doi.org/10.1109/tr.2024.3398409

[29] Singh, T. (2025). Digital Resilience, Cybersecurity and Supply Chains. Taylor & Francis.
[30] Somayajula, R. (2025). Hybrid AIOps for Resilient Data Pipelines: Integrating Statistical Analysis, Graph Intelligence, and Autonomous Agents for Proactive Maintenance [Review of Hybrid AIOps for Resilient Data Pipelines: Integrating Statistical Analysis, Graph Intelligence, and Autonomous Agents for Proactive Maintenance]. 2025 IEEE 6Th Global Conference for Advancement in Technology (GCAT), 1–8. https://doi.org/10.1109/gcat66372.2025.11368358
[31] Song, Z. (2025). Research on Implementation Pathways of AI-assisted Decision-making in Data Platform Architecture Optimization. Advances in Computer and Communication, 6(4), 236–243. https://doi.org/10.26855/acc.2025.10.014
[32] Souza, A. (2024). Observability and Monitoring [Review of Observability and Monitoring]. Tech Leadership Playbook, 171–191. https://doi.org/10.1007/979-8-8688-0543-1_7
[33] Tariq, F., Shaik, M. H., Mirza, S. B., & Islam, M. A. (2026). Service Failure Detection in Distributed Microservice Platforms. Saudi Journal of Engineering and Technology, 11(05), 501–510. https://doi.org/10.36348/sjet.2026.v11i05.013
[34] Tiwari, A. (2024). Unveiling Graph Structures in Microservices: Service Dependency Graph, Call Graph, and Causal Graph [Review of Unveiling Graph Structures in Microservices: Service Dependency Graph, Call Graph, and Causal Graph]. https://doi.org/10.59350/hkjz0-7fb09
[35] Ugwueze, V. U. (2024). Cloud Native Application Development: Best Practices and Challenges. International Journal of Research Publication and Reviews, 5(12), 2399–2412. https://doi.org/10.55248/gengpi.5.1224.3533
[36] Usanovich, G. Z. (2024). THE EVIDENCE OF AUDITING AND IMPORTANT ASPECTS OF THE DOCUMENTATION PROCESS. International Journal Of Management And Economics Fundamental, 4(6), 14–21. https://doi.org/10.37547/ijmef/volume04issue06-02
[37] Uy, O. (2016). Systems Reliability (pp. 366–397). Crc. https://doi.org/10.1201/b11291-16
[38] Velepucha, V., & Flores, P. (2023). A survey on microservices architecture: Principles, patterns and migration challenges. [Review of A survey on microservices architecture: Principles, patterns and migration challenges.]. IEEE Access, 11, 1–1. https://doi.org/10.1109/ACCESS.2023.3305687
[39] Vidyasagar Vangala. (2025). DevSecOps: Integrating Security into the DevOps Lifecycle. International Journal of Artificial Intelligence and Machine Learning, 16(01), 11–25. https://doi.org/10.5281/zenodo.15483597
[40] Weigand, H., & Johannesson, P. (2023). How to Identify your Design Science Research Artifact. 24, 1–10. https://doi.org/10.1109/cbi58679.2023.10187511
[41] Wendt, D. W. (2025). AI Governance and Risk Management (pp. 97–116). Apress. https://doi.org/10.1007/979-8-8688-1733-5_5
[42] Xia, B., Zhu, L., Lu, Q., Bi, T., & Xing, Z. (2023). On the Way to SBOMs: Investigating Design Issues and Solutions in Practice. In arXiv (Cornell University). European Society for Socially Embedded Technologies. https://doi.org/10.48550/arxiv.2304.13261
[43] Yeligeti, M., Gils, H. C., & Nowak, W. (2025). A composite metric for evaluating system resilience with non-idealistic performance curves. PloS One, 20(11), e0335909. https://doi.org/10.1371/journal.pone.0335909
[44] Yuan, F., & Lu, J. (2022). The Misconception and Risk of Using MTBF as a Reliability Indicator for Industry Management (pp. 324–332). Atlantis Bv. https://doi.org/10.2991/978-94-6463-038-1_30