Journal of Information Systems Engineering and Management
2026, 11(1s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Automation of Alerts Based on Operational Maturity Levels

Anshul Verma
Independent Researcher, USA

ARTICLE INFO

Received: 05 Nov 2025
Revised: 20 Dec 2025
Accepted: 01 Jan 2026

ABSTRACT

The automation of handling alerts in cloud native environments presents great opportunities for improvement in the area of operational efficiency. However, initial adoption without adequate process maturity creates the risk of cascading failures and loss of the trust of operators. Alert management systems have to attempt a delicate equilibrium between sensitivity and specificity in order to ensure that the critical events are detected with a minimum number of false positives, which contribute to alert fatigue. As has been previously discussed, without strong underlying mechanisms such as explicit service ownership, documented dependencies, and clear feedback loops in place, incident response effectiveness deteriorates due to automation since automated systems serve to magnify rather than rectify deficiencies inherent with lower-level machinery. To deploy automation in a successful manner, one should maturely pass through different stages: low risks in terms of data enrichment to augment human decision, to orchestration of workflow to eliminate/simplify coordination overhead, up to bounded autonomous response within controlled rotation of well-defined guardrails. Utilizing service-oriented patterns in integration architecture makes it possible to deploy automation across heterogeneous observability/deployment/service management platforms. Governance mechanisms: Approval hierarchies, kill switches, rate limiting, detailed audit logging, etc., keep automation in line with organizational goals but arbitrarily capped in safety terms. Automation readiness appears to correlate quite a bit more with the maturity of a process than with the capabilities of a technical infrastructure, placing automation firmly in the organizational capabilities camp, in which automation is gained through a planned, purposeful development of operational capabilities rather than technology simply deployed through investments in infrastructure.

Keywords: Alert Automation, Operational Maturity, Progressive Automation, Resilience Engineering, Service-Oriented Architecture

1. INTRODUCTION

Modern cloud-native operations present a peculiar landscape: while automation promises streamlining incident response and reducing operational burden, premature implementation contributes to the very problems automation is meant to solve. In practice, organizations that rush to implement automations around alert handling before establishing underlying processes are often met with increased noise, decreased operator trust, and cascading failures that spread more quickly than human teams can contain. Alert management systems are the critical interface between monitoring infrastructure and response teams, but research has shown that ineffective implementation results in alert fatigue, or the desensitization of operators to genuine critical events due to excessive notifications [1]. This is a reality that calls for a more sophisticated treatment: one that sees automation as a cultural achievement that is gained via maturity in an operation rather than merely introduced as a technology.

The difficulty isn't with the automation tools per se; rather, it's organizational readiness to embrace them. And so, when alerts automatically trigger responses in environments that do not have well-defined service boundaries, clear

Journal of Information Systems Engineering and Management
2026, 11(1s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

ownership, and mechanisms of systematic feedback, chaos masked as efficiency results. Alert management systems need to strike a balance between sensitivity, proportion of critical events that are converted into notifications, and specificity, minimizing false positives, which wear down operator attention [1]. Operational practices studies indicate that in installations characterized by poorly set alert thresholds, incident response effectiveness deteriorates significantly as personnel are swamped by volumes of notifications beyond their capacity to process. This is essentially a problem of deploying automation capability before the process maturity necessary to define meaningful alerting criteria, keep service dependency mappings up-to-date, and put feedback loops in place that continually tune the quality of alerts based on actual operational outcomes.

Conversely, organizations that pair automation strategies with operational maturity have the potential to unlock meaningful improvements in incident detection and recovery while ensuring the continued resilience and operator confidence of these systems. Continuous delivery pipelines in which automated testing, deployment, and monitoring are part of an integrated ecosystem highlight the integration of automation into DevOps practices [2]. When the maturity progression of automation deployment is followed, team confidence is the result of incremental capability expansion versus disruptive wholesale transformation. Continuous delivery methodologies demonstrate that automation succeeds when embedded within well-defined workflows where roles, responsibilities, and escalation paths have been explicitly established and socialized across operational teams [2]. The quantitative evidence strongly supports staged approaches where foundational processes—including incident classification frameworks, service ownership models, and post-incident review cadences—precede advanced automation capabilities.

However, what is the difference between successful and problematic automation initiatives? It's timing and scope. Progressive automation, which begins with data enrichment on low-risk matters, progresses through workflow orchestration to fully autonomous response with tight boundaries, enabling the Operational teams to build confidence in steps, and giving the muscle memory of the organization needed to sustain automation at scale. Alert management system research foregrounds how exemplary implementations are always located within a careful consideration of organizational context, including team structure, service complexity, and operational maturity level [1]. DevOps automation frameworks similarly stress that continuous improvement cycles, where automation performance is regularly assessed and refined based on operational feedback, represent essential components of sustainable automation strategies [2]. The evidence suggests that automation readiness correlates more strongly with process maturity metrics than with technical infrastructure capabilities, fundamentally reframing automation as an organizational capability rather than merely a technological deployment.

Alert System Characteristic	Operational Impact
Sensitivity balancing	Critical event notification
Specificity optimization	False positive minimization
Process discipline establishment	Meaningful alerting criteria
Service dependency mapping	Accurate dependency tracking
Feedback loop implementation	Continuous alert quality refinement
Incident classification frameworks	Foundation for automation
Service ownership models	Clear responsibility assignment
Post-incident review cadences	Systematic improvement cycles
DevOps integration	Interconnected ecosystem creation
Maturity-aligned deployment	Incremental capability expansion

Table 1: Impact of Alert System Configuration on Operational Effectiveness [1,2]

Journal of Information Systems Engineering and Management
2026, 11(1s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

2. THE MATURITY FRAMEWORK FOR ALERT AUTOMATION

2.1 Defining Operational Maturity Levels

A structured maturity model provides the foundation for responsible automation adoption. At the earliest level, organizations operate in an ad-hoc mode where manual triage dominates and incidents are handled reactively without standardized processes. As maturity progresses, teams establish consistent escalation patterns, define service ownership, and implement formal incident review cycles. Software reliability engineering research emphasizes that systematic approaches to quality management—including defect prevention, removal, and tolerance strategies—form the cornerstone of mature operational practices [3]. The shift from ad-hoc to managed processes involves the creation of repeatable workflows wherein incident response adheres to documented processes as opposed to the knowledge/skills/experience of individuals or even SME tribes. Organizations at initial maturity levels experience significant variability in resolution outcomes, as response effectiveness depends entirely on which personnel happen to be available during incident windows.

The intermediate stage introduces defined service level objectives and feedback loops that inform continuous improvement. Service availability metrics become central to operational governance, with targets typically ranging from 99.9% to 99.99% depending on service criticality and business requirements [4]. Alert prioritization becomes data-driven, with severity classifications mapped directly to business impact metrics and user-facing service degradation thresholds. Research on cloud computing availability demonstrates that achieving high reliability requires systematic attention to redundancy, fault tolerance, and proactive monitoring rather than reactive incident handling [4]. Organizations implementing structured SLO frameworks gain the ability to make informed trade-offs between development velocity and operational stability, using error budgets to guide release decisions and resource allocation priorities.

Higher maturity levels feature well-integrated tooling, automated enrichment of alert context, and eventually, autonomous response capabilities that operate within carefully defined boundaries. At these advanced stages, observability platforms automatically correlate alerts with recent deployment events, infrastructure changes, and historical incident patterns, reducing context-gathering time substantially. Software engineering approaches emphasize that reliability improvements require continuous measurement and feedback, with metrics guiding iterative refinement of both systems and processes [3]. Autonomous response systems at the highest maturity level operate with clear boundaries, executing well-validated remediation procedures for known failure modes while escalating novel or complex scenarios to human operators for assessment and resolution.

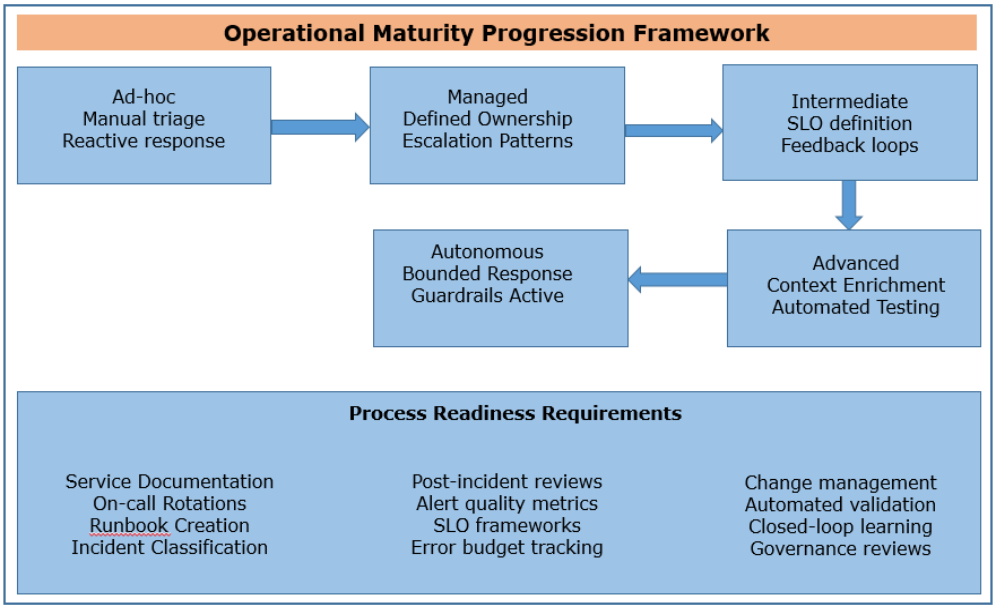

Figure 1: Operational Maturity Progression Framework [3,4]

Journal of Information Systems Engineering and Management
2026, 11(1s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

2.2 Process Readiness as a Prerequisite

Each maturity stage has its specific process prerequisites that must be laid down before automation can be successful. The early stages require well-documented services and their dependencies, definition of on-call rotations, well-defined responsibilities, and runbooks that encapsulate tribal knowledge. Middle stages require implementation of post-incident review processes, definition of actionable service level objectives, and development of alert quality metrics that guide refinement. Research on software reliability demonstrates that defect prevention through rigorous engineering practices proves more effective than post-deployment fault tolerance mechanisms, highlighting the importance of building quality into processes from inception [3]. Organizations must establish systematic review cycles where incident patterns inform preventive measures, alert tuning decisions, and architectural improvements.

Advanced stages build upon these foundations with sophisticated change management integration, automated testing of response procedures, and closed-loop learning systems that continuously optimize both alerts and automated responses. Cloud computing environments require particular attention to availability management, as distributed architectures introduce complex failure modes spanning multiple service boundaries and infrastructure layers [4]. Proactive monitoring capabilities are critical to the availability of cloud services in order to detect degradation before complete service failure, allowing for graceful degradation and rapid recovery. Mature organizations implement comprehensive testing regimens that validate normal operations as well as those expected during failures; automated responses are predictable under a wide range of conditions, including partial system failures, network partitions, and resource exhaustion scenarios.

Maturity Stage	Key Characteristics
Ad-hoc manual triage	Reactive incident handling without standardization
Managed escalation	Consistent patterns with defined ownership
Formal review cycles	Structured post-incident analysis
Service level objectives	Availability targets from 99.9% to 99.99%
Feedback loop integration	Continuous improvement mechanisms
Automated context enrichment	Alert correlation with deployment events
Autonomous remediation	Validated procedures for known failures
Advanced tooling integration	Cross-platform observability correlation
Change management sophistication	Automated testing of response procedures
Closed-loop learning systems	Continuous alert and response optimization

Table 2: Evolution of Process Capabilities Across Maturity Levels [3,4]

3. THE RISKS OF PREMATURE AUTOMATION

Deploying automation before achieving adequate maturity creates several critical failure modes. Alert fatigue intensifies when automated systems generate responses to poorly-tuned alerts, creating notification storms that overwhelm operators and mask genuine issues. Research on security operations demonstrates that automation misuse occurs when systems are deployed without adequate consideration of human factors, resulting in operators becoming overwhelmed by information volume or developing inappropriate reliance on automated recommendations [5]. Failure cascades accelerate when automated remediation actions trigger unintended side effects in systems where dependencies remain poorly understood or inadequately mapped. The cognitive burden imposed by excessive automation can paradoxically reduce situational awareness, as operators shift from active monitoring to passive supervision, degrading their ability to detect subtle anomalies or emerging patterns that fall outside automated detection parameters [5].

Perhaps most insidiously, trust erosion occurs when automation makes incorrect decisions repeatedly, causing operators to bypass or ignore automated systems entirely, thereby negating any potential benefits. Studies examining human-automation interaction reveal that automation disuse emerges when operators experience repeated instances where automated systems provide incorrect guidance or fail to detect genuine threats [5]. But,

the interaction between human and automation must be carefully balanced: automated systems operating with excessive amounts of automation lead to a debilitation of average skills among operators who are relegated to operating the system; but systems with insufficient automation lead to overworked staffs of manual operators. The biggest challenge in security Ops is a delicate balance between the advantages of automation and over-reliance: operators may not trust critical thinking to analyze an automated recommendation or, at the other extreme, reject valid automated recommendations due to the accumulated mistrust from incorrect suggestions in the past.

Organizations lacking established escalation governance particularly struggle with automation. Without clear ownership and accountability structures, automated actions can execute in ambiguous contexts where no human stakeholder feels responsible for outcomes. This creates a dangerous vacuum where systems take actions that no individual would approve, yet no process exists to prevent or review these decisions systematically. Resilience engineering research emphasizes that complex systems require adaptive capacity—the ability to respond effectively to unexpected situations that fall outside predefined response protocols [6]. Organizations implementing automation without adequate governance frameworks sacrifice this adaptive capacity, as automated systems lack the contextual understanding and judgment necessary to handle novel failure modes or situations involving conflicting objectives.

All of these risk factors act in a compounding way, creating negative reinforcement spirals. Alert fatigue causes the vigilance of operators to decrease, and there is a strong chance that automated errors will remain undetected for a longer time. Cascading failures compromise the stability of the system, generating further spurious alerts that put further strain on the monitoring function. Resilience engineering principles highlight that system safety emerges not from eliminating failures, but from developing organizational capabilities to detect, contain, and recover from inevitable disruptions [6]. Premature automation undermines these capabilities by introducing brittleness—automated responses optimized for expected scenarios may catastrophically fail when confronted with unanticipated conditions. The concept of graceful extensibility, where systems maintain performance even when operating outside design parameters, becomes critical for understanding automation risks [6]. Automated systems typically exhibit sharp performance boundaries, transitioning abruptly from effective operation to complete failure when encountering conditions beyond training data or programmed logic.

Breaking this cycle requires deliberate maturity development, addressing process gaps before expanding automation scope. Organizations must develop what resilience engineering terms "requisite variety"—sufficient diversity in response capabilities to match the complexity and variability of potential failure modes [6]. This includes maintaining human expertise alongside automated capabilities, ensuring that operators retain both the skills and authority to intervene when automated systems prove inadequate. Trust degradation motivates operators to disable or circumvent automation, eliminating potential benefits while leaving underlying alert quality issues unaddressed, creating organizational inertia that impedes future automation initiatives even after foundational issues have been resolved.

4. PROGRESSIVE AUTOMATION STRATEGY

4.1 Starting with Data Enrichment

Effective automation begins not with response actions but with information augmentation. Automatically enriching alerts with contextual data—recent deployments, related metrics, affected user segments, and relevant documentation—provides immediate value while introducing minimal risk. Cognitive systems engineering research emphasizes that human performance in complex systems depends fundamentally on information quality and accessibility, with decision-making effectiveness directly correlating to the availability of relevant contextual cues [7]. The foundational automation improves human decision-making without removing humans from critical decision points, building operator confidence and demonstrating clear value. Data enrichment reduces cognitive load by pre-assembling information that operators would otherwise need to gather manually from disparate sources, allowing mental resources to focus on diagnostic reasoning and strategic decision-making rather than information retrieval tasks.

Journal of Information Systems Engineering and Management
2026, 11(1s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Metric correlation provides another high-value, low-risk enrichment capability. Automatically attaching related performance indicators—such as CPU utilization, memory pressure, request latency distributions, and error rate trends—to alert notifications enables operators to distinguish between symptomatic alerts and root cause indicators. Cognitive systems engineering approaches recognize that effective performance emerges from the interaction between human cognitive capabilities and system design characteristics, suggesting that automation should augment rather than replace human expertise [7]. Documentation linkage further accelerates response, with automated attachment of relevant runbooks and architecture diagrams reducing time spent searching internal wikis. The principle of supporting rather than supplanting human judgment proves particularly important during initial automation phases, where operator acceptance depends on perceiving automation as a valuable assistant rather than a threatening replacement.

4.2 Advancing to Workflow Automation

Once data enrichment proves reliable and organizations have established strong process foundations, automation can extend to workflow orchestration. Automated creation of incident tickets, notification of appropriate teams based on service ownership, and assembly of response war rooms represent logical next steps. These automations handle coordination overhead while preserving human judgment for assessment and remediation decisions. Notification routing based on service ownership models ensures that incidents reach personnel with appropriate expertise and authority to respond effectively. Automated assembly of incident response channels—including relevant subject matter experts, service owners, and management stakeholders—accelerates collaborative troubleshooting during critical incidents. Integration with on-call scheduling systems ensures notifications reach available personnel, with automated failover to secondary responders when primary contacts remain unacknowledged.

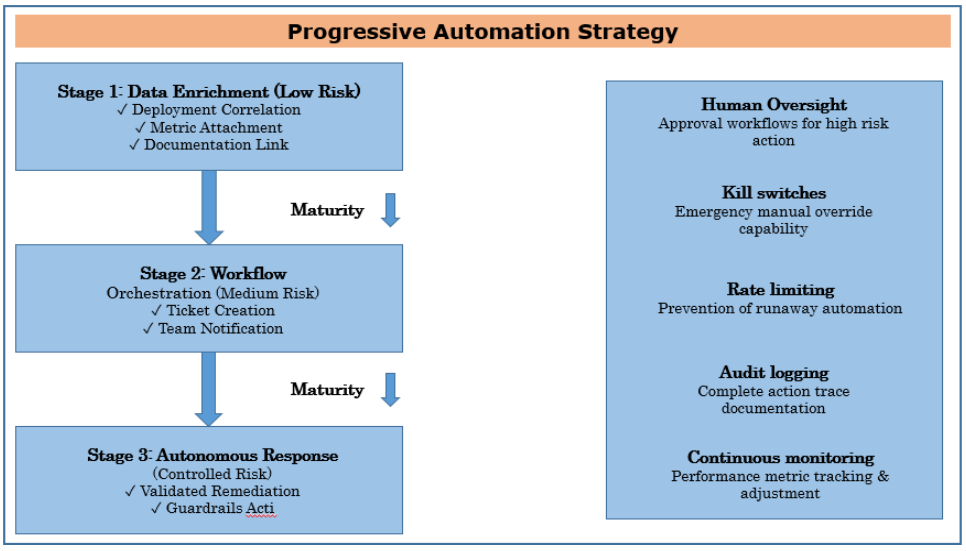

Figure 2: Progressive Automation Strategy [7,8]

4.3 Autonomous Response Within Boundaries

Full autonomous response emerges only at advanced maturity levels, and even then operates within carefully defined guardrails. Automated remediation applies to well-understood failure modes where runbooks have been extensively validated, rollback procedures are proven, and blast radius is limited. Research on organizational accident models demonstrates that system safety requires understanding how multiple contributing factors combine to produce failures, rather than assuming linear cause-and-effect relationships [8]. Human oversight remains embedded through approval workflows for high-risk actions and continuous monitoring of automation effectiveness. The Swiss cheese model of accident causation illustrates how defenses, barriers, and safeguards contain inherent weaknesses that occasionally align to permit failures, emphasizing the necessity of defense-in-depth approaches where multiple independent safety mechanisms provide redundancy [8].

Journal of Information Systems Engineering and Management
2026, 11(1s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Guardrail mechanisms limit autonomous actions to pre-approved change windows, restrict modifications to non-critical infrastructure components, and enforce maximum blast radius constraints. Rate limiting prevents runaway automation, with configurations allowing limited automated remediation attempts within defined time windows before escalating to human operators. Understanding accident causation through systemic models rather than simplistic linear thinking enables organizations to design automation with appropriate safety margins and failure containment mechanisms [8]. Continuous monitoring tracks automation effectiveness metrics, including success rate, rollback frequency, and time to recovery, with automatic suspension of autonomous capabilities when performance degrades below defined thresholds, ensuring that automation remains a net positive contributor to system reliability rather than introducing new failure modes.

Automation Capability	Implementation Characteristics
Contextual data augmentation	Recent deployments and related metrics attachment
Deployment event correlation	Automatic association with alert notifications
Performance metric integration	CPU, memory, latency trends inclusion
Documentation linkage	Runbook and architecture diagram attachment
Automated ticket creation	Incident record generation workflow
Service ownership routing	Team notification based on responsibility
War room assembly	Response channel with stakeholder inclusion
Validated runbook execution	Well-understood failure mode remediation
High-risk approval workflows	Human oversight for critical actions
Blast radius enforcement	Maximum impact constraint mechanisms

Table 3: Progressive Automation Implementation Stages [7,8]

5. IMPLEMENTATION CONSIDERATIONS

5.1 Integration Architecture

Successful automation requires thoughtful integration across observability platforms, deployment pipelines, and service management systems. Alert sources must provide rich, structured data rather than simple notifications. Research on service-oriented architecture demonstrates that IoT and cloud-based applications increasingly rely on modular, loosely coupled services that communicate through well-defined interfaces, enabling flexible integration patterns [9]. Deployment systems must expose change information that automation can correlate with incidents. Service management platforms must allow programmatic interaction while retaining audit trails and approval workflows where needed. Service-oriented architecture frameworks underline the necessity of standardized protocols of communication, message formats, and interface definitions for heterogeneous system interoperability without custom point-to-point integrations [9].

Authentication and authorization mechanisms enforcing the principle of least privilege restrict automated actors to specific operation scopes, with role-based access control models limiting the blast radius of compromised automation credentials to predefined service boundaries. Observability platforms implementing standardized instrumentation facilitate vendor-neutral telemetry collection, with distributed tracing capabilities enabling correlation of alerts across service boundaries. The architectural patterns practiced in service-oriented systems-message queuing, event-driven communication, and asynchronous processing-provide foundational capabilities that automation systems leverage to achieve reliability and scalability [9]. Time-series databases optimized for metric storage efficiently query historical data to enable real-time enrichment of alerts with contextual information derived from the operational history. Event streaming platforms offer durable message queues that ensure reliable delivery of alerts, even under heavy traffic situations that may arise during a large-scale incident. Architectural decisions have a direct effect on the extent of effectiveness and resilience of the automation.

5.2 Governance and Safety Mechanisms

Journal of Information Systems Engineering and Management
2026, 11(1s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

In other words, good governance is the assurance that the automation is aligned with organizational goals and that it is acceptable to face risks. Such considerations include setting approval hierarchies for the various types of automation, setting kill switches where automated actions can be manually overridden, rate capping to avoid runaway automation, or keeping full audit logs of all automated actions. System-Theoretic Accident Model and Processes research emphasizes that safety emerges from understanding control structures and constraints within complex sociotechnical systems, rather than from isolated component reliability [10]. Regular reviews assess whether automation boundaries remain appropriate as systems and requirements evolve. The STAMP approach recognizes that accidents result from inadequate control or enforcement of safety constraints, highlighting the necessity of governance mechanisms that continuously monitor and adjust automation behavior based on operational feedback and evolving risk profiles [10].

Kill switch mechanisms enable emergency suspension of automation, with circuit breaker patterns automatically disabling malfunctioning automation after consecutive failures within defined time windows. Rate-limiting configurations constrain automation to specified action frequencies per service, with burst allowances accommodating legitimate incident response scenarios while preventing runaway execution loops. Audit logging systems capture complete action traces, including timestamps, triggering conditions, execution parameters, and outcomes, supporting compliance requirements and post-incident analysis. System-theoretic approaches to safety analysis recognize that hazards arise not merely from component failures but from unsafe interactions between components, suggesting that automation governance must address both individual automation behaviors and their emergent system-level effects [10].

Automated testing of automation workflows themselves provides continuous validation, with synthetic incident injection verifying end-to-end functionality at regular intervals. Chaos engineering practices deliberately introduce controlled failures to validate automation behavior under degraded conditions, with experiments affecting limited subsets of non-production capacity to minimize operational impact while maximizing learning. Quarterly governance reviews examine automation effectiveness metrics, incident contributions, and false action rates, adjusting scope and guardrails based on accumulated operational evidence and evolving risk profiles. The STAMP methodology emphasizes that safety requires continuous attention to constraint enforcement across organizational, technical, and operational dimensions, aligning closely with the governance review cadences that maintain automation alignment with safety objectives over time [10].

Component Implementation	Technical or Governance Element
Modular service architecture	Loosely coupled communication interfaces
Protocol standardization	Uniform message format definitions
Event-driven messaging	Asynchronous processing patterns
Distributed tracing	Cross-service alert correlation
Time-series database optimization	Historical metric query efficiency
Multi-level approval hierarchies	Risk-based authorization workflows
Emergency kill switches	Immediate manual override capability
Action rate limiting	Runaway automation prevention
Comprehensive audit trails	Complete action trace logging
Quarterly governance reviews	Automation boundary reassessment

Table 4: Integration and Governance Architecture Components [9,10]

CONCLUSION

The path to effective automation of alerts is for organizations to treat automation as an earned capability rather than a technology bought, where the key to success is the maturity of operations rather than technical excellence. Premature introduction of automation has critical consequences for the liveability of the system because it worsens existing operational vulnerabilities that result in alert fatigue, distrust, and chaining of failures that aggravate rather than improve system reliability. Progressive implementation strategies can be enabled, such as starting from data enrichment and guiding through to workflow orchestration and ending in a bounded autonomous response, to

permit organizations to build automation progressively without fatigue to their operators or frailty to their systems. The maturity framework guides the decision-making by systematically determining the organizational readiness, open areas of the process that need filling before moving to expand the scope of automation, and governing mechanisms to keep automation transparent and aligned with the safety limits and organizational goals. So, an integration architecture, which is based on service-oriented design patterns, permits flexible deployment of automation this way throughout observability platforms, deployment pipelines, and service management systems, and without compromising audit trails and approval workflows. Governance mechanisms, such as kill switches, rate limiting, and continuous monitoring of effectiveness, are crucial mechanisms that stop automated systems from operating out of control and enable rapid manual intervention in cases where automated systems lack effectiveness. In order to achieve the necessary responsiveness, organizations will need to build requisite variety through human know-how, automated systems, and a combination of human and automated systems, tackling such failures and situations as being contradictory functions. In relation to automation, when it is embedded in well-defined processes that explicitly define roles, responsibilities, and escalation routes, it is a moving practice because it can be continuously refined in response to operational feedback and changing risk profiles based on continuous improvement loops.

REFERENCES

[1] Mrs. S.V.Gunthe et al., "SEnhancing Awareness: A Study on Alert Management Systems", IJAEM, 2024. [Online]. Available: https://ijaem.net/issue_dcp/SEnhancing%20Awareness%20A%20Study%20on%20Alert%20Management%20Systems.pdf

[2] Ravina Vijay Kahar et al., "Empowering DDevOps Enhancing Continuous Delivery Through Automation", IJCRT, Aug. 2025. [Online]. Available: https://www.ijcrt.org/papers/IJCRT2508225.pdf

[3] Aasia Quyoum et al., "Improving Software Reliability using Software Engineering Approach- A Review", International Journal of Computer Applications, 2010. [Online]. Available: https://www.ijcaonline.org/volume10/number5/pxc3871990.pdf

[4] Sanjay P Ahuja and Sindhu Mani, "Availability of Services in the Era of Cloud Computing", ResearchGate, 2012. [Online]. Available: https://www.researchgate.net/publication/268062088_Availability_of_Services_in_the_Era_of_Cloud_Computing

[5] Jack Tilbury and Stephen Flowerday, "Humans and Automation: Augmenting Security Operation Centers", MDPI, 2024. [Online]. Available: https://www.mdpi.com/2624-800X/4/3/20

[6] Klaus Thoma et al., "Resilience Engineering as Part of Security Research: Definitions, Concepts and Science Approaches", European Journal for Security Research - Springer Nature, 2016. [Online]. Available: https://link.springer.com/article/10.1007/s41125-016-0002-4

[7] Dana Rad et al., "A Cognitive Systems Engineering Approach Using Unsupervised Fuzzy C-Means Technique, Exploratory Factor Analysis and Network Analysis—A Preliminary Statistical Investigation of the Bean Counter Profiling Scale Robustness", MDPI, 2022. [Online]. Available: https://www.mdpi.com/1660-4601/19/19/12821

[8] Fabrizio Bracco and Martina Ivaldi, "How Metaphors of Organizational Accidents and Their Graphical Representations Can Guide (or Bias) the Understanding and Analysis of Risks", MDPI, 2023. [Online]. Available: https://www.mdpi.com/2079-3200/11/10/199

[9] Joao Giao et al., "A Framework for Service-Oriented Architecture (SOA)-Based IoT Application Development", MDPI, 2022. [Online]. Available: https://www.mdpi.com/2227-9717/10/9/1782

[10] Riccardo Patriarca et al., "The past and present of System-Theoretic Accident Model And Processes (STAMP) and its associated techniques: A scoping review", ScienceDirect, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0925753521004082