International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 12-No.1 (2026) pp. 1107-1116
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

AI-Assisted Incident Response: Engineering Safety into Automated Operations

Sumit Kaul*

Independent Researcher, USA
* **Corresponding Author Email:** reachsumitkaul@gmail.com - **ORCID:** 0000-0002-5247-1150

Article Info:

DOI: 10.22399/ijcesen.4976
Received : 29 December 2025
Revised : 20 February 2026
Accepted : 22 February 2026

Keywords

AI Copilots,
Incident Response,
Site Reliability Engineering,
Safety Guardrails,
Human-AI Collaboration

Abstract:

Modern distributed systems present unprecedented challenges for incident response, with telemetry volumes and architectural complexity overwhelming human cognitive capacity during critical outages. This article examines the integration of large language models as copilots for incident management, proposing a comprehensive framework that balances the speed advantages of artificial intelligence with rigorous safety controls. The article identifies three critical failure points in incident response—sense-making across disparate telemetry sources, hypothesis generation under stress, and safe mitigation execution—where AI assistance shows promise but also introduces significant risks, including hallucination, privilege boundary violations, and lack of production constraint awareness. Drawing on frameworks for AI risk management, software supply chain security, and human-AI collaboration, the article presents a three-phase architecture separating sensing, deciding, and acting with mandatory human validation gates between transitions. The proposed multi-layer safety framework encompasses data governance through automated redaction and schema validation, privilege architecture implementing separation of duties and risk budgets, verification mechanisms including counterfactual checking and shadow execution, and comprehensive auditability through immutable decision ledgers. Human-AI collaboration patterns emphasize augmentation rather than replacement of human judgment, with AI providing rapid data synthesis and pattern matching while humans contribute contextual reasoning, ethical judgment, and final decision authority. The framework demonstrates that bounded automation with explicit oversight can reduce detection and restoration times while preserving the reliability guarantees and accountability requirements that production systems demand, offering organizations a practical path to leveraging AI assistance without compromising operational safety.

1. Introduction

OpenTelemetry has emerged as a critical framework for observing and monitoring modern software systems, providing developers with comprehensive insights into application behavior and performance. According to research by Malte Hansen et al. published in "Instrumentation of Software Systems with OpenTelemetry for Software Visualization" on ResearchGate in November 2024, the framework represents a significant advancement in how software systems can be instrumented for visualization purposes. The research emphasizes that OpenTelemetry offers a vendor-neutral, open-source approach to collecting telemetry data including traces, metrics, and logs from software applications.The framework addresses fundamental challenges in distributed systems where understanding the flow of requests across multiple services becomes increasingly complex. As detailed in the publication from ScienceDirect titled "Software visualization: A systematic literature review," software visualization techniques have evolved to help developers comprehend intricate system architectures and identify performance bottlenecks. The integration of OpenTelemetry with visualization tools creates powerful capabilities for developers to trace requests through their entire lifecycle, from initial user interaction through various microservices and back to the response.

Hansen et al. explain in their ResearchGate publication that the instrumentation process involves embedding OpenTelemetry components within application code to automatically capture telemetry data without requiring extensive manual

coding. This automatic instrumentation significantly reduces the development overhead traditionally associated with monitoring implementations. The framework supports multiple programming languages and integrates seamlessly with existing applications, making it accessible for diverse technology stacks. The collected data encompasses distributed traces that map request paths, metrics that quantify system performance, and logs that provide contextual information about application events.

The visualization aspect discussed in both references highlights how raw telemetry data transforms into actionable insights through graphical representations. According to the ScienceDirect systematic literature review, effective software visualization helps developers identify patterns, anomalies, and optimization opportunities that might otherwise remain hidden in vast amounts of log data. When combined with OpenTelemetry's standardized data collection, visualization tools can present comprehensive views of system health, dependency relationships, and performance characteristics across distributed architectures.

The practical implications of this instrumentation approach extend beyond simple monitoring. Hansen et al. emphasize that OpenTelemetry-based visualization enables teams to perform root cause analysis more efficiently, understand system dependencies with greater clarity, and make informed architectural decisions based on actual runtime behavior. The framework's extensibility allows organizations to customize data collection and visualization according to their specific needs while maintaining compatibility with industry-standard observability platforms. This flexibility ensures that as systems evolve and scale, the instrumentation infrastructure adapts accordingly, providing consistent visibility into increasingly complex software ecosystems.

2. The Operational Challenge

Incident response fails at three critical junctions, each representing a distinct cognitive and operational burden on human responders. First, sense-making across disparate telemetry sources overwhelms individual responders who must mentally integrate logs, performance metrics, distributed traces, recent deployments, and service dependencies. The transition to microservices architectures has fundamentally altered the landscape of operational complexity, introducing challenges that traditional monolithic system management practices were not designed to address. Research by Soldani, Tamburri, and Van Den Heuvel examining the operational implications of microservices through a systematic grey literature review reveals that while these architectures offer benefits in terms of deployment flexibility and team autonomy, they simultaneously create significant observability and debugging challenges [3]. The distributed nature of microservices means that a single business transaction can traverse multiple service boundaries, with each service maintaining its own logging configuration, metric collection strategy, and deployment lifecycle. This fragmentation makes it substantially more difficult for operators to reconstruct the complete picture of system behavior during incidents, as they must correlate evidence across heterogeneous telemetry sources that may use different timestamping schemes, aggregation intervals, and semantic conventions [3]. The cognitive overhead of maintaining mental models of these distributed interactions while simultaneously processing incoming alert signals represents a fundamental bottleneck in incident response effectiveness.

Second, generating plausible root cause hypotheses requires pattern recognition across historical incidents and current symptoms—a task hampered by stress and incomplete information. The hypothesis generation phase demands both systematic analysis of observable symptoms and intuitive recognition of patterns that match historical failure modes. In microservices environments, this challenge intensifies because failures can manifest through complex interaction patterns that are not immediately obvious from examining individual service metrics. A performance degradation in one service might result from resource contention in a shared infrastructure component, cascading timeout effects from a downstream dependency, or subtle changes in request patterns originating from upstream callers. Studies of microservices operational practices by Soldani and colleagues indicate that the architectural complexity introduced by service decomposition creates scenarios where root cause analysis requires expertise spanning multiple team boundaries, as the services involved in a failure path may be owned by different teams with distinct operational practices and monitoring approaches [3]. This organizational fragmentation compounds the technical challenges of diagnosis, as responders may lack direct access to telemetry from all affected services or may need to coordinate with multiple teams to gather necessary diagnostic information.

Third, executing safe mitigations demands procedural discipline: rolling back deployments, adjusting rate limits, or failing over traffic requires

understanding blast radius and reversibility. Research on automated diagnosis frameworks for distributed systems by Demirbaga and colleagues demonstrates both the potential and the limitations of machine-driven remediation approaches. Their AutoDiagn framework, designed for real-time anomaly detection and automated diagnosis in big data systems, can process streaming telemetry to identify performance degradations and correlate them with specific system components or recent changes [4]. These frameworks employ sophisticated algorithms to analyze metrics across multiple dimensions, detecting deviations from expected behavior patterns and generating diagnostic hypotheses about likely root causes. The AutoDiagn system achieves this through a three-tier architecture combining metric collection agents, statistical analysis engines, and decision-making components that work in concert to identify anomalies within seconds of their occurrence [4]. However, the transition from diagnosis to remediation introduces significant risks, as automated actions must account for complex dependencies, capacity constraints, and potential side effects that may not be fully captured in the system's model of its own behavior. Even well-designed automated remediation systems require careful calibration of action thresholds and comprehensive testing to ensure they do not inadvertently worsen incidents through inappropriate or poorly timed interventions [4].

Language models excel at cross-domain pattern matching and summarization, making them natural candidates for these tasks. Yet they suffer from fundamental limitations: they hallucinate nonexistent patterns, cannot inherently understand privilege boundaries, and lack situational awareness about production constraints. The design question becomes not whether AI can assist, but how to structure its participation to maximize value while minimizing harm.

3. Architectural Foundations

The software supply chain has become increasingly vulnerable to sophisticated attacks, prompting the development of comprehensive security frameworks like Supply-Chain Levels for Software Artifacts, commonly known as SLSA. According to research by Mahzabin Tamanna et al. published in "Unraveling Challenges with Supply-Chain Levels for Software Artifacts SLSA for Securing the Software Supply Chain" on ResearchGate in September 2024, SLSA represents a critical advancement in establishing security standards for software artifacts throughout their development lifecycle. The framework provides a structured approach to mitigate risks associated with compromised dependencies, malicious code injection, and unauthorized modifications that can occur at various stages of software production.

The complexity of modern software supply chains necessitates robust security measures, as highlighted in the ScienceDirect publication on supply chain security. Contemporary software applications typically incorporate numerous third-party libraries, open-source components, and dependencies that create multiple potential entry points for attackers. Tamanna et al. emphasize that SLSA addresses these vulnerabilities by implementing a progressive security model with multiple levels of assurance, each building upon the previous level to create increasingly stringent security requirements. This graduated approach allows organizations to incrementally improve their security posture while accommodating varying resource capabilities and risk tolerances.

The research published on ResearchGate details specific challenges organizations face when implementing SLSA frameworks in real-world environments. These challenges include the technical complexity of establishing comprehensive build provenance, maintaining detailed audit trails across distributed development teams, and ensuring verification mechanisms function correctly across heterogeneous toolchains. Tamanna et al. explain that achieving higher SLSA levels requires significant investment in infrastructure automation, continuous monitoring systems, and developer training programs. Organizations must balance the security benefits against implementation costs and potential impacts on development velocity.

According to the ScienceDirect publication, effective supply chain security requires end-to-end visibility into how software artifacts are created, modified, and distributed. SLSA accomplishes this through cryptographically signed attestations that document the entire build process, from source code to final deliverable. The framework mandates specific security controls including isolated build environments, two-person review processes for critical changes, and automated verification of dependency integrity. These requirements create an auditable trail that enables organizations to detect tampering attempts and trace the provenance of every component within their software ecosystem.

The practical implications of SLSA adoption extend beyond immediate security improvements. Tamanna et al. note that organizations implementing SLSA frameworks often discover operational efficiencies and improved development practices as byproducts of enhanced transparency and automation. However, the research also acknowledges persistent challenges including the

need for industry-wide standardization, interoperability between different security tools, and the ongoing maintenance burden of keeping security attestations current as software evolves. As supply chain attacks continue to increase in frequency and sophistication, frameworks like SLSA become essential components of comprehensive cybersecurity strategies, providing structured pathways for organizations to strengthen their defenses against emerging threats.

4. Multi-Layer Safety Framework

4.1 Data Governance

In that respect, input hygiene prevents the model from reasoning over compromised or inappropriate data, setting foundational controls to protect both the AI system and the organization from potential privacy violations and security breaches. Systems strip personally identifiable information via automated detection and redaction pipelines, finding patterns matching names, email addresses, phone numbers, and other identifiable attributes before telemetry reaches the model, masking secrets including API keys, authentication tokens, database credentials, and cryptographic material that might appear in log streams or configuration snapshots. Instead of forwarding complete log streams that can contain thousands of entries with redundant or sensitive information, these systems sample the logs intelligently, selecting representative examples that capture error patterns and diagnostic context while minimizing data volume and exposure risk. Research into privacy vulnerabilities in machine learning systems indicates critical concerns around membership inference attacks, where adversaries can determine whether specific data records were included in a model's training set by looking at the model's behavior on those records. These attacks involve the tendency of machine learning models to memorize aspects of their training data, with the potential for models to accidentally expose sensitive information even when the model is designed for some other primary purpose [7]. The risk of such attacks underscores the importance of data minimization in AI systems: models that never process sensitive information cannot inadvertently memorize or leak it through their outputs. Only structured artifacts with source attribution enter into the reasoning pipeline, rejecting free-form text pastes without provenance that could introduce unverified or malicious content. Temporal boundaries ensure recommendations draw from recent snapshots rather than stale data, with preprocessing pipelines automatically filtering out observations older than defined thresholds to prevent the AI from basing decisions on outdated system state.

4.2 Privilege Architecture

Separation of duties prevents the copilot from approving or deploying changes autonomously, implementing a core security principle that distributes critical authorities across multiple actors to prevent unilateral action. The copilot drafts proposals tagged to responsible owners who retain decision authority, ensuring that domain experts with accountability for affected systems exercise final judgment on all modifications. Each suggestion carries a risk budget defining permissible scope in terms of affected services or infrastructure components, duration specifying maximum time windows for changes to remain active, and magnitude quantifying the scale of modification, such as percentage of traffic affected or number of configuration parameters altered—anything exceeding these bounds requires human escalation to senior engineers or incident commanders. The model accesses only an allow-listed toolset, typically read-only dashboards for querying current system state and synthetic checks for validating hypotheses about system behavior, never direct production system access that would enable state modification. Studies examining membership inference attacks demonstrate that even seemingly benign model access patterns can enable sophisticated adversaries to extract training data characteristics, suggesting that any AI system with access to sensitive operational data requires rigorous privilege controls that limit both what data the system can access and what operations it can perform with that data [7]. This defense-in-depth approach recognizes that individual controls may fail or be bypassed, making layered redundancy essential for maintaining safety guarantees.

4.3 Verification Mechanisms

Before accepting any suggestion, the system performs counterfactual checking: the copilot must state what evidence would disprove its hypothesis, and the orchestrator retrieves that evidence to challenge the proposal. High-risk mitigations undergo shadow execution in staging environments that replicate production topology and traffic patterns, with automated rollback triggers monitoring key performance indicators. Critical changes require dual approval from distinct roles, preventing single points of failure in decision-making. Research on formal verification techniques for deep neural networks addresses the fundamental

challenge of ensuring that AI systems behave safely across their entire input space. Traditional software testing validates correctness on finite test suites, but neural networks operate in high-dimensional continuous spaces where exhaustive testing is impossible. Formal verification approaches attempt to prove mathematical properties about network behavior, such as demonstrating that a network will produce outputs within specified bounds for all inputs in a defined region [8]. These techniques employ constraint-solving methods to systematically explore the network's decision boundaries, identifying inputs that could trigger undesirable behaviors [8]. While computational complexity limits verification to relatively small networks, the principles inform practical testing strategies that combine targeted adversarial examples with statistical sampling to gain confidence in AI system reliability.

4.4 Auditability Requirements

Every interaction is captured in an immutable decision ledger recording inputs, prompt versions, model outputs, approvals, and measured outcomes. Summaries must cite specific signals—trace identifiers, monitor names, deployment tags—enabling rapid verification and post-incident analysis. This evidence trail supports both real-time validation and long-term system improvement.

5. Human-AI Collaboration Patterns

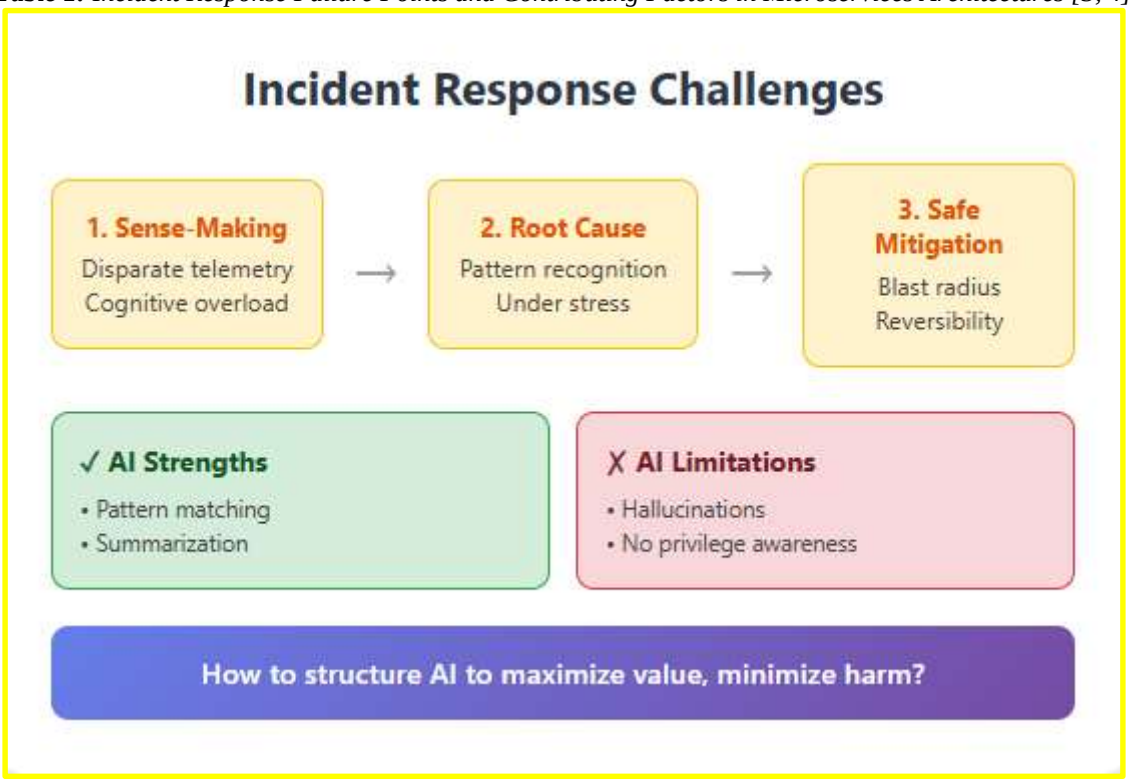
Effective copilots augment rather than replace human judgment, embodying a collaborative paradigm where artificial intelligence serves as a decision support tool rather than an autonomous agent. Within the first minutes of an incident, the system produces a situational brief identifying blast radius through analysis of service dependency graphs and current traffic patterns, likely regression points by correlating symptom onset timing with recent deployment or configuration changes, and top hypotheses ranked by confidence scores derived from similarity to historical incident patterns alongside immediate low-risk checks such as verifying monitoring system health or confirming that alert correlation logic is functioning correctly. Research by Aman and colleagues examining trends in human-AI collaboration through systematic review reveals that effective partnership between humans and AI systems requires careful attention to interface design, task allocation, and trust calibration [9]. Their analysis demonstrates that AI systems achieve optimal impact when they complement human cognitive capabilities rather than attempting to replicate them entirely, with successful implementations focusing on scenarios where AI can process large data volumes rapidly while humans contribute contextual reasoning and ethical judgment. The field has evolved from early automation-focused approaches that sought to replace human decision-makers toward more nuanced models that recognize the irreplaceable value of human expertise, particularly in high-stakes operational contexts where decisions carry significant consequences and require accountability that only humans can provide [9].

For each hypothesis, the copilot drafts verification steps that specify exactly what telemetry queries or synthetic tests would confirm or refute the proposed explanation, mitigation sequences that outline ordered actions with explicit dependencies and prerequisites, and success criteria that define observable metrics or system behaviors that would indicate successful remediation. Humans can accept these proposals and proceed directly to execution, modify them to incorporate institutional knowledge or context not available to the AI system, or reject them entirely in favor of alternative approaches based on their expertise and situational judgment—all with minimal friction through streamlined user interfaces that present options clearly and enable rapid selection. The interaction design reflects principles from human-centered AI research by Camara and colleagues, which emphasizes the critical importance of maintaining human agency and control throughout AI-assisted workflows [10]. Their framework for human-machine interaction in automated vehicles identifies design principles that enable effective collaboration, including making AI capabilities and limitations clear to users so they can develop appropriate mental models of system behavior, supporting efficient correction mechanisms when AI outputs require modification, and scoping AI services to well-defined tasks where success criteria can be clearly specified and measured. These guidelines recognize that users interact with AI systems across diverse contexts and with varying levels of expertise, necessitating designs that adapt to user needs while maintaining consistent interaction patterns that reduce cognitive load [10].

Post-incident, the copilot generates preliminary review outlines linking timeline events such as initial symptom detection, hypothesis formulation, mitigation attempts, and resolution confirmation to ledger entries that document the evidence and reasoning behind each decision point, which responders refine into final documentation suitable for organizational learning and compliance requirements. The integration of AI-generated draft documentation with human editorial refinement exemplifies collaborative approaches that leverage

the comparative advantages of both human and artificial intelligence. Research on human-machine interaction emphasizes that AI systems should provide timely information relevant to user goals without overwhelming them with unnecessary details, and should facilitate efficient dismissal or refinement of AI suggestions that do not align with user intent or situational requirements [10]. The post-incident review process benefits from AI's ability to comprehensively catalog events and correlate them with system telemetry, while humans provide interpretive analysis that connects technical details to organizational context, identifies systemic improvement opportunities, and ensures that documentation serves its intended purposes for knowledge transfer and compliance demonstration. This division of labor reflects a mature understanding of collaborative intelligence, where neither humans nor AI attempt to subsume the other's role but instead work in concert to achieve outcomes neither could accomplish independently [9].

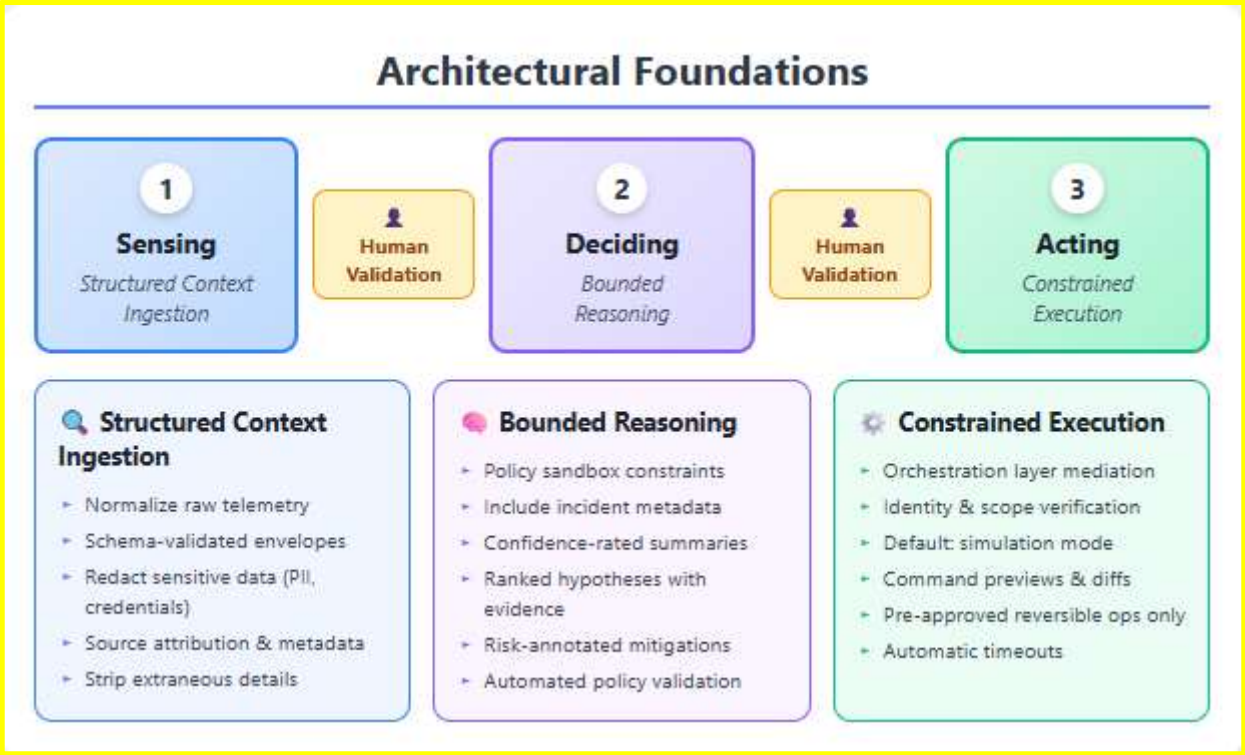
Table 1: *Incident Response Failure Points and Contributing Factors in Microservices Architectures [3, 4]*



Failure Junction	Primary Challenge	Technical Complexity Factor	Organizational Impact	AI Solution Potential	Risk Without Guardrails
Sense-Making	Integrating disparate telemetry (logs, metrics, traces)	Heterogeneous timestamping schemes and semantic conventions across services	Cognitive bottleneck in processing multiple alert signals	Cross-domain pattern matching and rapid data synthesis	Hallucination of nonexistent correlations
Hypothesis Generation	Pattern recognition across historical and current incidents	Complex interaction patterns spanning multiple service boundaries	Requires expertise across team boundaries with different monitoring approaches	Historical pattern matching and similarity detection	Overgeneralization from incomplete information

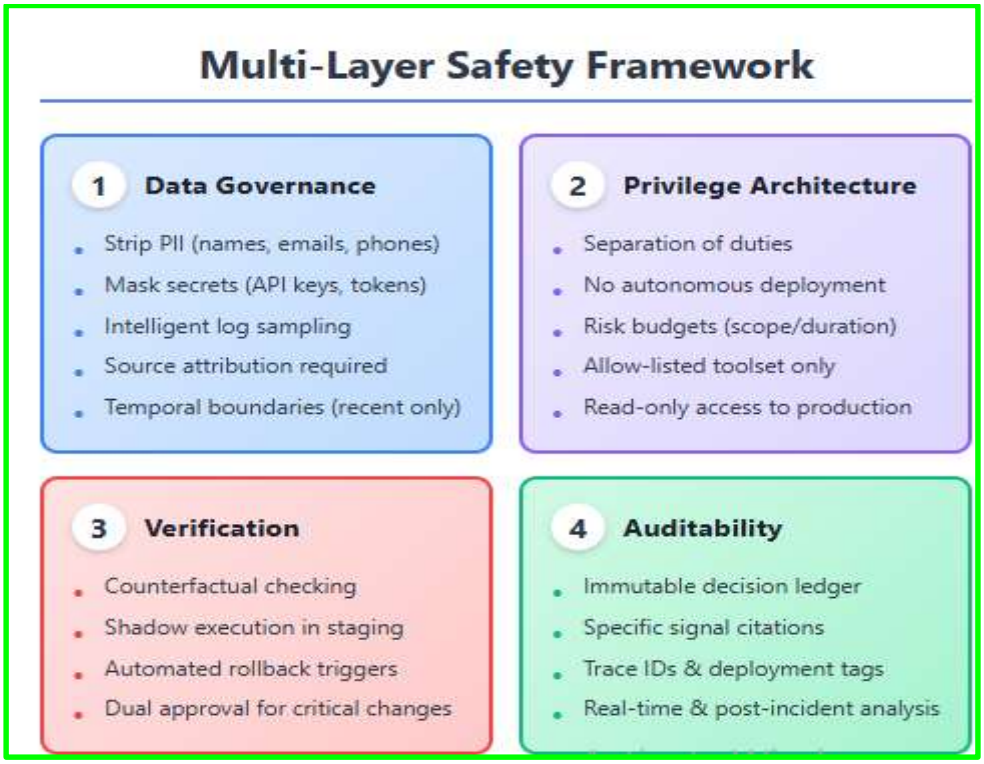
Mitigation Execution	Understanding blast radius and reversibility of actions	Complex dependencies and capacity constraints	Coordination requirements for cross-team service changes	Automated action proposal with risk annotation	Lack of privilege boundaries and production constraint awareness

Table 2: *Data Processing and Security Controls Across Copilot Architecture Phases [5, 6]*



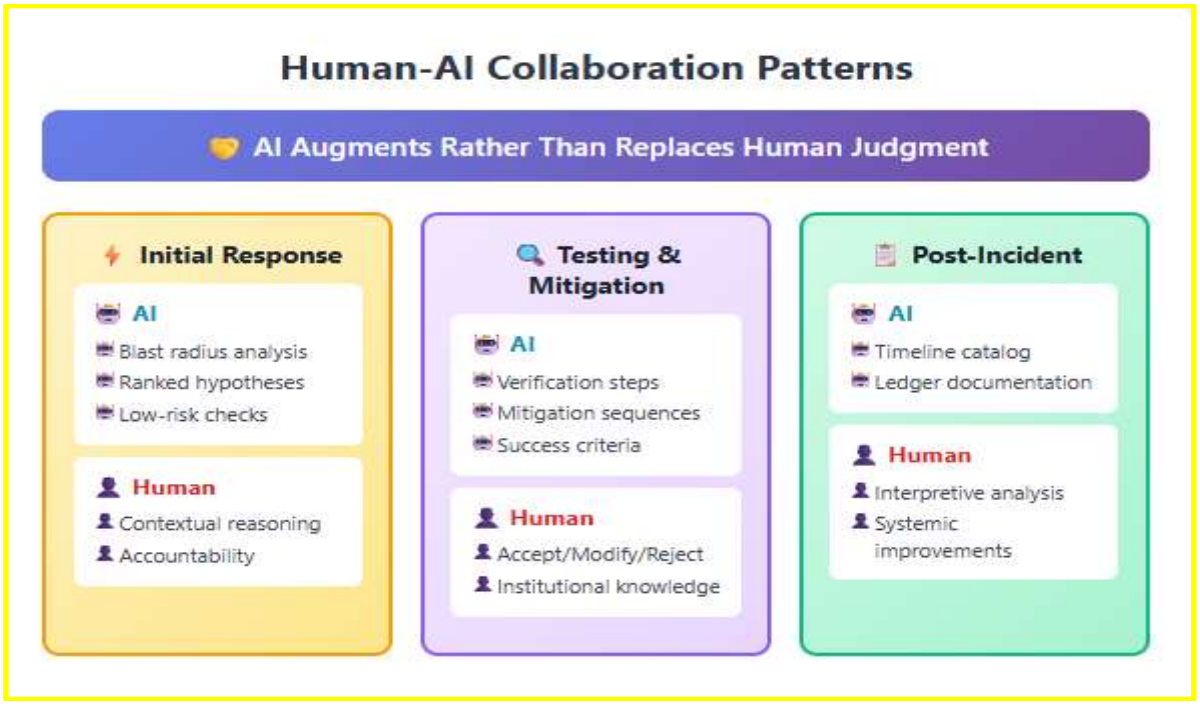
Architecture Component	Data Transformation	Privacy Control	Security Boundary	Risk Mitigation Strategy	Validation Requirement
Normalization Pipeline	Transform to schema-validated envelopes with standardized structures	Strip PII, mask credentials, sample logs	Data preparation layer enforces access controls	Automated redaction prevents sensitive data exposure	Schema compliance verification
Policy Sandbox	Structured prompts with incident metadata and operational constraints	Context limited to essential information only	Computational constraints prevent unauthorized exploration	Risk tolerance boundaries encoded in sandbox rules	Confidence threshold validation
Orchestration Layer	Generate executable commands or configuration changes	No additional data processing	Identity verification and scope enforcement	Verifiable provenance tracking mirrors supply chain integrity	Dual authorization for critical changes
Execution Environment	Apply approved modifications to production systems	Audit logging of all actions	Change windows and automatic timeouts	Default simulation mode with explicit approval gates	Post-execution outcome monitoring

Table 3: *Multi-Layer Safety Framework Overview [7, 8]*



Safety Layer	Primary Control	Threat Mitigated	Key Mechanism
Data Governance	Automated redaction and sampling	Privacy violations and data exposure [7]	Strip PII, mask secrets, validate schemas
Privilege Architecture	Separation of duties	Autonomous unauthorized actions	Risk budgets, allow-listed tools, and owner approval
Verification Mechanisms	Counterfactual checking	Hallucinated hypotheses and unsafe changes [8]	Challenge proposals, shadow execution, dual approval
Auditability Requirements	Immutable decision ledger	Untrackable actions	Record all inputs, outputs, and approvals with citations

Table 4: *Task Allocation and Complementary Strengths in Human-AI Collaboration [9, 10]*



Capability Type	AI Strength	Human Strength	Collaboration Principle	Design Guideline
Data Processing	Rapid processing of large data volumes [9]	Contextual reasoning and pattern recognition	AI complements rather than replicates human cognition	Make AI capabilities and limitations clear
Pattern Recognition	Similarity matching to historical incidents	Intuitive recognition of novel failure modes	AI handles routine patterns; humans address unique cases	Support efficient correction mechanisms
Decision Making	Generate ranked options with confidence scores	Ethical judgment and accountability	AI proposes; humans decide	Maintain human agency and control
Documentation	Comprehensive event cataloging and correlation	Interpretive analysis and organizational context	AI provides structure; humans add meaning	Provide timely information without overwhelming details
Execution Authority	Draft proposals with risk annotations	Final approval and situational judgment	Neither subsumes the other's role	Scope AI to well-defined tasks

6. Conclusions

Large language models can fundamentally transform incident response from reactive firefighting to evidence-driven operations when safety is systematically engineered into their deployment architecture. The article presented demonstrates that effective AI copilots require rigorous implementation of four foundational layers: structured context ingestion that normalizes heterogeneous telemetry and enforces privacy boundaries, bounded reasoning within policy sandboxes that constrain solution spaces to organizational risk tolerance, constrained execution with identity verification and scope limits that prevent unauthorized modifications, and comprehensive auditability through immutable decision ledgers that enable both real-time validation and continuous improvement. Success depends on recognizing that AI systems serve as decision support tools rather than autonomous agents, complementing human cognitive capabilities in data processing and pattern matching while humans retain authority over interpretation, judgment, and execution. The collaboration model emphasizes transparent presentation of AI capabilities and limitations, efficient mechanisms for human correction and override, and clear task scoping where success criteria can be objectively measured. Organizations must treat copilot systems themselves as governed infrastructure components subject to versioning, validation, and rollback procedures equivalent to any critical production system. The standard of operation is clear: AI systems should suggest clearly and with evidence sources, only operate within limited pre-approved limits, be subject to constant control in terms of counterfactual, shadow execution, and record all interactions comprehensively to facilitate accountability and learning. Within these limitations, AI copilots will be true force multipliers, enhancing human knowledge, lessening cognitive load in high-stress events, and speeding up a solution without diminishing the reliability and trust that modern platforms promise their end users, which will allow organizations to achieve operational excellence, none other than in scale and complexity that modern distributed systems necessitate.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

[1] N Kavyashree et al., "Site reliability engineering for IOS mobile application in small-medium scale industries," ScienceDirect, November 2021. Available: https://www.sciencedirect.com/science/article/pii/S2666285X21000935

[2] Malte Hansen et al., "Instrumentation of Software Systems with OpenTelemetry for Software Visualization," ResearchGate, November 2024. Available: https://www.researchgate.net/publication/385944956_Instrumentation_of_Software_Systems_with_OpenTelemetry_for_Software_Visualization

[3] Jacopo Soldani et al., "The pains and gains of microservices: A systematic grey literature review," Dec. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0164121218302139

[4] Umit Demirbaga et al., "AutoDiagn: An automated real-time diagnosis framework for big data systems,"Oct.-Dec. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9394788

[5] Kun Tian et al."Artificial intelligence in risk management within the realm of construction projects: A bibliometric analysis and systematic literature review," ScienceDirect, June 2025. Available: https://www.sciencedirect.com/science/article/pii/S2444569X25000617

[6] Mahzabin Tamanna et al., "Unraveling Challenges with Supply-Chain Levels for Software Artifacts SLSA for Securing the Software Supply Chain," ResearchGate, September 2024. Available: https://www.researchgate.net/publication/383911133_Unraveling_Challenges_with_Supply-Chain_Levels_for_Software_Artifacts_SLSA_for_Securing_the_Software_Supply_Chain

[7] Hongsheng Hu et al., "Membership Inference Attacks on Machine Learning: A Survey," ResearchGate, March 2021. Available: https://www.researchgate.net/publication/350088342_Membership_Inference_Attacks_on_Machine_Learning_A_Survey

[8] Xiaowei Huang et al, "Safety Verification of Deep Neural Networks," ResearchGate, July 2017. Available: https://www.researchgate.net/publication/318370372_Safety_Verification_of_Deep_Neural_Networks

[9] Attila Kovari, "A systematic review of AI-powered collaborative learning in higher education: Trends and outcomes from the last decade," Dec. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590291125000622

[10] Tintin Jiang et al., "Human-AI interaction research agenda: A user-centered perspective," December 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2543925124000147