International Journal of Multidisciplinary Sciences and Technology
ISSN: 3107-9911 | Volume 2 Issue 1, 34-41, January-March 2026
DOI: https://doi.org/10.64137/31079911/IJMST-V2I1P105

Received: 19/01/2026 Revised: 21/01/2026 Accepted: 25/01/2026 Published: 29/01/2026

Original Article

Designing Scalable AI Systems for Continuous Monitoring, Fault Detection, and Operational Intelligence

[1]DR. ANANYA PILLAI, [2]DR. VIKRAM DESHMUKHX [3]DR. HARSHA BHATX, [4]DR. SNEHA RAJPUTX, [5]DR.TEJASWINI RAO
[1]Department of Artificial Intelligence, Central Institute of Machine Intelligence Assistant Professor Thiruvananthapuram, India.
[2]Department of Computer Science, Riverdale University of Technology Assistant Professor Nagpur, India.
[3]Department of Information Technology, Peninsula Institute of Information Sciences Assistant Professor Mangaluru, India.
[4]Department of Computer Science, Institute of Digital Innovation and Systems Assistant Professor Indore, India.
[5]Department of Artificial Intelligence Academy of Emerging AI Studies Assistant Professor Visakhapatnam, India.

ABSTRACT: Modern digital platforms are expected to operate continuously across heterogeneous clouds, microservices, data pipelines, AI-enabled applications, and cyber-physical workflows. In such environments, conventional monitoring is no longer sufficient because it reports isolated symptoms instead of generating system-level understanding. This paper proposes an architecture-centered research framework for scalable AI systems that unify continuous monitoring, fault detection, and operational intelligence into a closed-loop capability. The framework integrates telemetry collection, semantic data normalization, learning-based anomaly detection, graph-based dependency reasoning, and policy-aware decision intelligence so that systems can move from reactive alerting to evidence-based adaptation. The paper synthesizes advances in observability engineering, log intelligence, software defect prediction, automated testing, vulnerability detection, and domain-specific operational analytics to define a layered reference architecture suitable for software-intensive and distributed environments. Rather than presenting fabricated benchmark outcomes, the paper contributes a research-grade design blueprint, a model taxonomy, and an evaluation agenda for future empirical validation. The framework is intentionally cross-domain: it can support cloud-native software systems, healthcare and pharmacy operations, customer AI platforms, financial systems, manufacturing, and supply-chain networks. The paper argues that scalable operational intelligence emerges when telemetry, prediction, and action are treated as a single architectural problem rather than as disconnected toolchains.

KEYWORDS: AIOps, observability, fault detection, operational intelligence, microservices, anomaly detection, decision intelligence, software reliability, trace analytics, cloud-native AI systems.

1. INTRODUCTION

The operational profile of modern enterprise systems has changed significantly. Business workflows now span microservices, APIs, event streams, AI models, distributed storage, customer-facing applications, and external dependencies that evolve independently. A system can appear healthy at the infrastructure layer while still degrading at the transaction, feature, or decision layer. As a result, reliability engineering increasingly depends on end-to-end telemetry and learning-based interpretation rather than on static threshold alarms alone [1][2][3]. Recent work on customer AI observability similarly emphasizes that model behavior, feature drift, and inference outcomes must be traced across system boundaries if organizations want to understand why decisions succeed or fail in production [4]. Continuous monitoring and fault detection are often implemented as separate disciplines. Monitoring teams focus on data collection, dashboards, and alert rules, while quality engineering teams focus on defects, testing, and release validation. Root cause analysis is then treated as a downstream incident task. This separation is structurally inefficient in distributed systems because failure signals propagate across logs, metrics, traces, topology changes, and business events at the same time. Graph-based dependency models show that faults rarely remain local; they spread through service call relations and shared infrastructure, making symptom-based diagnosis insufficient [5]. Log intelligence research has therefore shifted toward learning normal execution patterns from telemetry streams so that abnormal sequences can be detected earlier and more contextually [6][9][16][19].

At the same time, software engineering research continues to demonstrate the value of machine learning for defect prediction and quality-risk estimation before runtime failures occur [8]. Comparative studies of automated testing in enterprise Java systems and governance-centered frameworks for software-intensive platforms indicate that the best outcomes arise when predictive quality signals are fed into lifecycle decisions rather than examined in isolation [7][10][12]. Domain-specific studies in banking, cloud data pipelines, healthcare automation, customer engagement systems, and manufacturing further suggest that operational intelligence should not be limited to failure detection; it must also prioritize remediation, compliance, throughput, cost, and human impact [13][11][15][18][20]. This paper addresses that broader problem. It proposes a scalable AI systems architecture in which telemetry capture, feature engineering, anomaly detection, fault localization, and decision support are composed into a closed feedback loop. The paper makes four contributions. First, it synthesizes heterogeneous research strands into a unified reference architecture for continuous monitoring and operational intelligence. Second, it defines a model

taxonomy spanning log-based sequence learners, graph-based propagation analysis, defect prediction models, testing intelligence, and policy-aware decision layers. Third, it explains how such a system can be operationalized across distinct sectors including healthcare, finance, customer AI, cloud platforms, and manufacturing. Fourth, it outlines an evaluation framework that future empirical work can use to compare scalability, timeliness, accuracy, and actionability. This is an architecture and synthesis paper rather than an experimental benchmark study. Accordingly, its value lies in clarifying design principles, integration points, and research directions for practitioners and researchers building next-generation AIOps platforms. The central argument is straightforward: scalable AI operations require a system that not only senses and predicts faults, but also interprets business context and orchestrates the next best action.

2. RELATED WORK AND RESEARCH FOUNDATIONS

Observability has matured from a tooling concept into an architectural discipline. Industrial survey evidence shows that microservice tracing and analysis are now essential for understanding highly dynamic service interactions, especially when debugging performance regressions and cascading failures [3]. End-to-end observability for customer AI extends this principle by connecting data lineage, feature generation, and prediction outcomes, thereby making operational faults and model-level failures jointly visible [4]. These developments indicate that observability data must be semantically aligned if it is to support learning systems rather than merely retrospective inspection. Log-based anomaly detection has become a major research area because logs encode execution behavior with higher semantic resolution than coarse metrics. DeepLog demonstrated that sequence models can learn normal log-key transitions and identify deviations that correspond to anomalous execution paths [6]. Transformer-based approaches such as LogBERT improved contextual representation, especially when long-range dependencies matter [9]. Yet evaluation studies caution that performance claims in log anomaly detection are sensitive to data grouping, class distribution, and experimental assumptions, meaning that strong benchmark results do not always transfer to live systems [16][19]. This matters for production AI systems, where concept drift, workload seasonality, and noisy labels are common.

A second line of work concerns quality intelligence before deployment. Studies comparing machine learning models for software defect prediction show that predictive risk scoring can identify failure-prone components and prioritize testing effort [8]. Reliability-oriented analyses of automated testing in Java enterprise systems demonstrate that defect prevention is not only a modeling problem but also a process integration problem: predictive signals must influence test selection, release gates, and engineering workflows [7]. Governance-centered frameworks similarly argue that lifecycle decisions should be architecture-aware and continuously informed by predictive evidence [2][10][12]. From this perspective, runtime fault detection and pre-deployment quality assurance are parts of the same reliability continuum. Root cause localization research has evolved from simple dependency traversal toward multi-signal reasoning. Practical trace-analysis approaches exploit span-level relationships to identify likely root-cause services in microservice systems [14]. Causal discovery methods go further by modeling how failures propagate through service interaction graphs and how hidden causes may be inferred from observed performance variables [22]. Graph-based service-dependency models reinforce the importance of explicit topology for predicting failure propagation in distributed environments [5]. Together, these studies suggest that AI systems for operations should combine sequence intelligence with structural reasoning rather than relying on either alone.

Cross-domain research broadens the operational scope of such systems. Predictive monitoring for change data capture pipelines highlights how data freshness, replication lag, and schema variance become operational risks in cloud-native data platforms [11]. Secure microservices research in healthcare underscores that reliability must be coupled with privacy, compliance, and deployment portability [15]. Domain studies on prescription automation, OCR quality, and secure data-in-transit anomaly detection show that operational intelligence can blend defect detection with domain-specific accuracy and security goals [21][24][27]. Similarly, work on patient journey analytics, blockchain traceability, sustainable manufacturing, digital doubles, and global MES rollout demonstrates that observability and intelligent monitoring are increasingly relevant in socio-technical environments where software, human operators, and physical assets interact [20][18][23][26][37]. The research gap is therefore not the absence of capable individual models. The gap is the lack of a unifying architecture that can coordinate telemetry semantics, predictive reasoning, root-cause inference, and decision support across domains and scales. This paper fills that gap through a layered design framework.

3. A REFERENCE ARCHITECTURE FOR SCALABLE AI OPERATIONS

The proposed architecture is organized into five interacting layers: telemetry acquisition, semantic data modeling, intelligence services, operational decisioning, and adaptive governance. Its purpose is not to replace existing observability stacks, but to transform them into learning systems that generate timely and actionable operational knowledge.

TABLE 1 Proposed architecture layers for continuous monitoring and operational intelligence

Layer	Primary inputs	AI/analytics role	Operational output
Telemetry acquisition	logs, metrics, traces, events, topology, test results, security signals	capture, timestamp alignment, signal correlation	unified multi-signal event stream

Semantic data modeling	parsed telemetry, metadata, dependency graphs, ownership context	schema normalization, feature extraction, graph construction	feature store and dynamic system graph
Intelligence services	time-series windows, log sequences, code and release metadata	anomaly detection, defect prediction, drift detection, root-cause inference	ranked incidents, risk scores, explanations
Operational decisioning	alerts, predictions, policies, business priorities	prioritization, recommendation, automation trigger selection	remediation plans, routing, runbook actions
Adaptive governance	outcomes, feedback, postmortems, cost and SLA data	model recalibration, threshold tuning, policy learning	updated models, controls, and assurance evidence

3.1. TELEMETRY ACQUISITION

The foundation of the architecture is complete yet economical telemetry collection. Metrics alone are insufficient because they reveal magnitude but not execution intent. Logs expose event semantics, and traces reveal causal flow across services. The system should therefore collect all three signals together, along with deployment events, test outcomes, configuration changes, and relevant business events. Survey evidence from industrial microservice environments shows that the usefulness of observability depends heavily on the ability to correlate signals across teams and components [3]. In AI-enabled products, the acquisition layer must also record feature versions, model identifiers, prompt templates where relevant, and prediction outputs so that model failures and software failures can be examined jointly [4].

3.2. SEMANTIC DATA MODELING

Raw telemetry is not directly suitable for AI reasoning. It must be normalized into semantically stable entities such as services, endpoints, instances, workflows, datasets, and releases. This layer performs log parsing, trace stitching, metric windowing, and topology assembly. Graph-based system representations are critical because distributed failures propagate along dependency paths rather than organizational boundaries [5]. The graph should be dynamic and time-aware, allowing the system to represent transient service calls, autoscaling effects, and deployment-specific dependencies. Semantic alignment also enables domain extensions: for instance, a healthcare pipeline may attach prescription identifiers and compliance tags, while a manufacturing system may include work-cell states, product lots, or carbon-footprint events [15][18][23].

3.3. INTELLIGENCE SERVICES

The intelligence layer hosts multiple model families instead of a single universal detector. Sequence models analyze logs for abnormal event progression [6][9]. Time-series detectors inspect latency, throughput, error rate, and resource metrics. Defect prediction models score code or services according to estimated fault proneness before release [8]. Automated testing intelligence maps those risks to targeted validation effort [7][13]. Structural inference models use graphs and traces to localize likely root causes and infer propagation chains [14][22]. The key design principle is complementarity: log models are strong at temporal pattern recognition, graph models are strong at structural reasoning, and defect models are strong at early risk estimation. A scalable system should fuse their outputs rather than force one model family to solve all tasks.

3.4. OPERATIONAL DECISIONING

Operational intelligence begins where anomaly detection ends. A mature platform should rank incidents not only by technical severity but also by business criticality, customer impact, compliance risk, recovery cost, and confidence level. This is where decision intelligence frameworks become essential [2][12]. For example, a moderate anomaly in a revenue-critical customer flow may deserve higher priority than a severe anomaly in a low-impact internal batch job. Similarly, a model-drift event in a regulated healthcare workflow may require manual review even when the fault is statistically mild [21][24]. The decisioning layer therefore uses learned severity estimates together with explicit policy constraints.

3.5. ADAPTIVE GOVERNANCE

Closed-loop improvement requires feedback. After each alert, remediation, or postmortem, the system should capture whether the diagnosis was correct, whether the action helped, what the cost was, and how long recovery took. This outcome evidence is used to recalibrate thresholds, retrain models, and update decision policies. Governance-oriented frameworks in software lifecycle management emphasize the importance of auditable, architecture-aware control loops rather than opaque automation [10][12]. In practice, this means that every automated recommendation should be explainable enough for humans to validate and refine.

4. INTELLIGENT MONITORING AND FAULT DETECTION MECHANISMS

A research-grade AI operations platform requires diversity in detection mechanisms because faults do not manifest uniformly. Some appear as abrupt metric spikes, others as slow performance drifts, sequence violations, test regressions, or security anomalies. The proposed framework therefore combines five complementary mechanisms.

4.1. LOG-SEQUENCE INTELLIGENCE

Log streams often preserve the earliest signs of failure. Sequence-learning methods remain valuable because they can identify unexpected execution transitions before aggregate metrics have materially changed. DeepLog introduced a practical and influential formulation of this idea by predicting next log events from preceding sequences [6]. Transformer-based approaches such as LogBERT improved contextual representation, especially when long-range dependencies matter [9]. However, empirical work warns that deployment realism matters: systems differ in log quality, parser stability, and anomaly prevalence, so models should be evaluated under drift, imbalance, and noisy segmentation assumptions [16][19]. Accordingly, the architecture should support continuous parser validation, online recalibration, and human review of anomalous templates.

4.2. GRAPH AND TRACE REASONING

When an anomaly is detected, the next challenge is to determine where it started and how it spread. Trace-based root cause localization can identify suspicious services by analyzing span relationships and invocation patterns [14]. Causal-discovery approaches help distinguish likely causes from correlated symptoms, which is especially useful in microservice systems with fan-out communication and latent dependencies [22]. Dynamic service-dependency graphs further improve reasoning by encoding the paths along which faults propagate [5]. In production use, graph reasoning should be updated continuously because static topology assumptions fail under autoscaling, canary releases, and ephemeral infrastructure.

4.3. PREDICTIVE QUALITY AND TESTING INTELLIGENCE

Not all faults should be discovered in production. Defect prediction studies show that model-based risk estimation can prioritize modules and services likely to contain faults [8]. Risk-aware testing frameworks in banking and enterprise systems indicate that such predictions become operationally useful only when they influence automated test selection, environment readiness checks, and release approvals [7][13]. The architecture therefore links pre-deployment quality signals with runtime monitoring. A service that carries high predicted defect risk can be instrumented more aggressively, assigned tighter rollout policies, or placed under enhanced watch during post-release stabilization.

4.4. SECURITY-AWARE AND COMPLIANCE-AWARE MONITORING

Operational intelligence is incomplete if it ignores privacy, security, and data integrity. Secure microservices for healthcare illustrate how architecture, deployment environment, and compliance rules jointly shape acceptable operational behavior [15]. Cloud-native prescription automation and OCR improvement studies further show that model accuracy failures can have direct operational and clinical consequences [21][24]. Research on anomaly detection and encryption strategies for fax communication demonstrates that data-in-transit risks can also be observed as operational anomalies, not just cybersecurity events [27]. Therefore, the system should treat security telemetry and compliance evidence as first-class inputs to anomaly scoring and remediation policy.

4.5. DOMAIN-AWARE CONTEXT INFUSION

The same anomaly can have different meanings in different domains. In patient-journey analytics, an inference delay may reduce engagement quality [20]. In manufacturing traceability, a telemetry discrepancy may threaten provenance assurance across supply chains [18]. Sustainable manufacturing platforms require visibility into sensor-driven process states and environmental metrics [23], while cloud-native CPG decision systems must connect operational data to planning decisions [32]. Human-robot collaboration and psychosocial digital doubles add another layer in which operator condition, workflow rhythm, and machine coordination influence what constitutes abnormal behavior [26][29]. In human-facing platforms, affective or interaction signals may also become relevant contextual features when operational quality depends on user state and engagement [17]. For this reason, scalable AI operations should expose domain-context adapters that enrich generic telemetry with sector-specific semantics.

5. FROM FAULT DETECTION TO OPERATIONAL INTELLIGENCE

The central difference between fault detection and operational intelligence is actionability. Detection asks whether something abnormal is happening. Operational intelligence asks what should be done next, by whom, at what urgency, and with what expected trade-off. This paper frames that transition through three design principles: prioritization, orchestration, and learning.

5.1. PRIORITIZATION

A ranked incident queue should integrate technical evidence with business relevance. Observability events from customer AI systems may need to be prioritized by model impact or downstream decision criticality [4]. Manufacturing environments may prioritize anomalies that threaten traceability or production continuity [18][37]. Data pipelines may prioritize freshness and delivery guarantees [1][11]. Digital-finance ecosystems similarly require prioritization logic that accounts for portfolio sensitivity, customer engagement, and real-time market-facing workflows [25]. In each case, the same underlying anomaly detector can feed a different decision policy because priorities differ by context. A robust platform therefore separates detection scores from business-priority policies while allowing the two to interact.

5.2. ORCHESTRATION

Operational intelligence must map a diagnosis to an intervention. Some faults are suitable for automatic rollback, traffic shifting, cache invalidation, or retry policy changes. Others require human escalation or a gated workflow due to regulatory or reputational risk. Research on lifecycle governance supports this view by framing decisions as evidence-driven, architecture-aware, and automation-sensitive [2][10][12][30]. In practice, the orchestration engine should output a recommendation object containing the likely root cause, confidence level, blast radius, candidate remediation steps, and whether automation is permitted.

5.3. LEARNING FROM OUTCOMES

A decision is only intelligent if the platform learns from its consequences. If a recommended rollback repeatedly resolves incidents for a specific service pattern, policy confidence should increase. If an anomaly class generates frequent false positives, thresholds or features should be adjusted. Redis-backed fulfillment optimization, cloud-platform comparisons, and deployment portability studies all imply that operational quality is influenced by infrastructure choices and workload-specific feedback [31][35]. Feedback-driven adaptation is therefore the mechanism by which an observability system becomes an operational intelligence system.

6. SCALABILITY, DEPLOYMENT, AND EVALUATION

A scalable AI operations platform must be architected for both computational scale and organizational scale. Computationally, it should support streaming ingestion, hierarchical storage, asynchronous feature generation, and cost-aware retention. Organizationally, it should support team ownership boundaries, policy variation, auditability, and gradual automation adoption. Model-serving infrastructure matters. Optimization studies in neural architecture search and learning dynamics suggest that model efficiency, convergence behavior, and numerical stability affect whether sophisticated detectors remain practical under production constraints [33][36][38]. Comparative analyses of software defect prediction models also remind us that model choice should reflect data characteristics and operational goals rather than default library popularity [28]. Infrastructure comparisons across cloud platforms likewise show that portability, runtime isolation, and deployment ergonomics influence operational resilience [35]. Consequently, the intelligence layer should separate online lightweight inference from offline heavy retraining, with explicit latency budgets for each operational path.

The evaluation of such a system should go beyond detector accuracy. A research-grade assessment should measure: 1. detection timeliness: how quickly abnormal behavior is identified after onset; 2. localization quality: whether the ranked root-cause candidates are correct and early enough to matter; 3. actionability: whether the output supports faster or better remediation; 4. business alignment: whether prioritization reflects cost, customer impact, or compliance exposure; 5. adaptation quality: whether the system improves after feedback; 6. operational cost: telemetry overhead, storage use, inference latency, and analyst burden.

TABLE 2 Suggested evaluation dimensions for future empirical studies

Dimension	Example measures	Why it matters
Detection performance	precision, recall, F1, mean time to detect	establishes analytical validity
Localization quality	top-k hit rate, mean reciprocal rank, explanation usefulness	links anomalies to actionable causes
Decision quality	mean time to acknowledge, mean time to recover, rollback success rate	measures operational value
Scalability	ingestion throughput, inference latency, cost per signal volume	determines feasibility at enterprise scale
Governance and trust	audit completeness, override rate, false-automation rate	ensures human and regulatory acceptability
Domain outcome	SLA adherence, data freshness, safety/compliance events, engagement quality	connects system intelligence to real-world impact

Cross-domain empirical validation is also important. Ensemble fault detection studies and neural architecture comparisons in software quality research suggest that no single model dominates in all settings [34][39]. Vulnerability detection work reinforces the need to integrate code-level features and representation learning when security faults are part of the operational picture [40]. Thus, future experiments should compare model families across software services, data pipelines, healthcare workflows, and industrial systems rather than assuming that benchmark performance generalizes automatically.

7. CHALLENGES AND FUTURE RESEARCH DIRECTIONS

Despite strong progress, several research challenges remain open. First, semantic heterogeneity continues to limit generalization. Logs, traces, business events, and model metadata follow different schemas across organizations. Without stable semantic conventions, transfer learning and cross-system benchmarking remain difficult. Second, ground truth is scarce.

Many incidents have incomplete labels, mixed causes, or evolving symptoms, which makes supervised training and fair evaluation challenging. Third, explainability is still immature in operational AI. Incident responders need ranked evidence, dependency paths, and counterfactual reasoning, not just anomaly scores. Causal methods and graph reasoning offer promise, but they remain sensitive to data quality and hidden confounders [14][22]. Fourth, closed-loop automation introduces governance risks. A platform that can automatically reroute traffic, suppress alerts, or roll back services must be policy-constrained and auditable to avoid harmful actions. Fifth, sector-specific adaptation remains underexplored. The same architecture can support healthcare, finance, customer AI, manufacturing, and cyber-physical systems, but each domain imposes distinct constraints around privacy, safety, latency, and human oversight [15][20][32][26]. Future research should therefore prioritize semantic standardization, multi-signal foundation models for operations, feedback-aware causal inference, and benchmark suites that include business-context labels rather than only technical anomaly labels. Another promising direction is the fusion of software engineering intelligence with runtime telemetry, allowing predicted defect proneness, vulnerability likelihood, and test outcomes to dynamically shape monitoring policy. Such integration would finally connect the software lifecycle with live operational adaptation rather than treating them as separate worlds.

8. CONCLUSION

Scalable AI systems for continuous monitoring, fault detection, and operational intelligence should be designed as integrated socio-technical platforms rather than as collections of disconnected observability tools and machine learning models. This paper proposed a layered research architecture that unifies telemetry acquisition, semantic modeling, intelligence services, operational decisioning, and adaptive governance. It drew on literature spanning observability, log anomaly detection, defect prediction, root-cause localization, security-aware monitoring, and domain-specific operational analytics. The resulting framework emphasizes complementarity across model types, explicit policy integration, and outcome-driven learning. Its broader implication is that reliability in modern distributed systems is no longer only about finding faults faster. It is about turning operational data into trustworthy decisions that improve resilience, efficiency, and human oversight across complex digital environments.

REFERENCES

[1] V. K. R. Mittamidi, “An Automated AI-Driven Monitoring and Observability Framework for Cloud-Based Data Pipelines by Software Defect Prediction Research,” International Journal of Multidisciplinary Evolutionary Research, vol. 5, no. 1, pp. 109–112, 2024, doi: https://doi.org/10.54660/ijmer.2024.5.1.109-112.

[2] S. D. Sivva, R. R. Thalakanti, S. S. G. Bandari, and S. D. R. Yettapu, “AI-Driven Decision Intelligence for Agile Software Lifecycle Governance: An Architecture-Centered Framework Integrating Machine Learning Defect Prediction and Automated Testing,” International Journal of Emerging Trends in Computer Science and Information Technology, vol. 4, pp. 167–172, 2023, doi: https://doi.org/10.63282/3050-9246.ijetcsit-v4i4p118.

[3] B. Li et al., “Enjoy your observability: an industrial survey of microservice tracing and analysis,” Empirical Software Engineering, vol. 27, no. 1, Nov. 2021, doi: https://doi.org/10.1007/s10664-021-10063-9.

[4] A. K. K. V. Alluri, “End-to-End Observability for Customer AI: Tracing Data, Features, and Predictions Across Systems,” Global Multidisciplinary Perspectives Journal, vol. 1, no. 5, pp. 67–70, 2024, doi: https://doi.org/10.54660/gmpj.2024.1.5.67-70.

[5] N. Mutyam, “Graph-Based Modeling of Service Dependencies for Predicting Failure Propagation in Distributed Systems,” International Journal of Multidisciplinary Evolutionary Research, vol. 5, no. 1, pp. 113–116, 2024, doi: https://doi.org/10.54660/ijmer.2024.5.1.113-116.

[6] M. Du, F. Li, G. Zheng, and V. Srikumar “DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning,” Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285-1298, 2017, doi: https://doi.org/10.1145/3133956.3134015

[7] S. R. Gudi, “Enhancing Reliability in Java Enterprise Systems through Comparative Analysis of Automated Testing Frameworks,” International Journal of Emerging Trends in Computer Science and Information Technology, vol. 4, 2023, doi: https://doi.org/10.63282/3050-9246.ijetcsit-v4i2p115.

[8] S. K. Gunda, “Analyzing Machine Learning Techniques for Software Defect Prediction: A Comprehensive Performance Comparison,” 2024 Asian Conference on Intelligent Technologies (ACOIT), pp. 1–5, Sep. 2024, doi: https://doi.org/10.1109/acoit62457.2024.10939610.

[9] H. Guo, S. Yuan, and X. Wu, “LogBERT: Log Anomaly Detection via BERT,” 2021 International Joint Conference on Neural Networks (IJCNN), Jul. 2021, doi: https://doi.org/10.1109/ijcnn52387.2021.9534113.

[10] S. D. R. Yettapu, “A Unified Artificial Intelligence Governance and Reliability Engineering Framework for Secure and Autonomous Software-Intensive and Cyber-Physical Systems,” Journal of Frontiers in Multidisciplinary Research, vol. 4, no. 1, pp. 605–608, 2023, doi: https://doi.org/10.54660/.jfmr.2023.4.1.605-608.

[11] V. K. R. Mittamidi, “Leveraging AI and ML for Predictive Monitoring and Error Mitigation in Change Data Capture Pipelines,” International Journal of Emerging Trends in Computer Science and Information Technology, vol. 6, pp. 104–111, 2025, doi: https://doi.org/10.63282/3050-9246.ijetcsit-v6i3p116.

[12] M. Balerao, "A Converged Artificial Intelligence Architecture for Innovation, Software Lifecycle Optimization, and Cybersecurity Risk Mitigation," International Journal of Multidisciplinary Futuristic Development, vol. 4, no. 1, pp. 117–120, 2023, doi: https://doi.org/10.54660/ijmfd.2023.4.1.117-120.

[13] S. K. Gunda, "A Risk-Aware AI Framework for Automated Testing and Quality Assurance in Core Banking Systems," International Journal of Multidisciplinary Evolutionary Research, vol. 5, no. 1, pp. 117–120, 2024, doi: https://doi.org/10.54660/ijmer.2024.5.1.117-120.

[14] Z. Li et al., "Practical Root Cause Localization for Microservice Systems via Trace Analysis," Jun. 2021, doi: https://doi.org/10.1109/iwqos52092.2021.9521340.

[15] S.R. Gudi, "Design and Evaluation of Secure Microservices Architecture for HIPAA-Compliant Prescription Processing on AWS and OpenShift," International Journal of Artificial Intelligence, Data Science, and Machine Learning, vol. 5, no. 2, Jun. 2024, doi: https://doi.org/10.63282/3050-9262.ijaidsml-v5i2p116.

[16] M. Landauer, S. Onder, F. Skopik, and M. Wurzenberger, "Deep learning for anomaly detection in log data: A survey," Machine Learning with Applications, vol. 12, p. 100470, Jun. 2023, doi: https://doi.org/10.1016/j.mlwa.2023.100470.

[17] GV Krishna, BD Reddy, and T. Vrindaa, "EmoVision: An Intelligent Deep Learning Framework for Emotion Understanding and Mental Wellness Assistance in Human Computer Interaction," International Journal of Artificial Intelligence, Data Science, and Machine Learning, vol. 6, 2025, doi: https://doi.org/10.63282/3050-9262.ijaidsml-v6i4p103.

[18] Prahlad Chowdhury, "BLOCKCHAIN FOR MANUFACTURING TRACEABILITY: SECURING MANUFACTURING DATA IN MULTI-TIER SUPPLY CHAINS," International Journal of Applied Mathematics, vol. 38, no. 11s, pp. 336–357, Nov. 2025, doi: https://doi.org/10.12732/ijam.v38i11s.1169.

[19] V.-H. Le and H. Zhang, "Log-based anomaly detection with deep learning," Proceedings of the 44th International Conference on Software Engineering, May 2022, doi: https://doi.org/10.1145/3510003.3510155.

[20] A. K. K. Varma Alluri, "Using Salesforce CRM and Deep Learning (CNN) Techniques to Improve Patient Journey Mapping and Engagement in Small and Medium Healthcare Organizations," International Journal of Artificial Intelligence, Data Science, and Machine Learning, vol. 6, 2025, doi: https://doi.org/10.63282/3050-9262.ijaidsml-v6i4p115.

[21] S. R. Gudi, "AI-Driven Fax-to-Digital Prescription Automation: A Cloud-Native Framework Using OCR, Machine Learning, and Microservices for Pharmacy Operations," International Journal of Emerging Research in Engineering and Technology, vol. 5, no. 1, Mar. 2024, doi: https://doi.org/10.63282/3050-922x.ijeret-v5i1p113.

[22] Azam Ikram et al., "Root Cause Analysis of Failures in Microservices through Causal Discovery," 36th Conference on Neural Information Processing Systems (NeurIPS 2022), vol. 35, pp. 31158-31170, 2022.

[23] P. Chowdhury, "Sustainable Manufacturing 4.0: Tracking Carbon Footprint In SAP Digital Manufacturing With IOT Sensor Networks," Frontiers in Emerging Computer Science and Information Technology, vol. 2, no. 9, pp. 12–19, Sep. 2025, doi: https://doi.org/10.37547/fecsit/volume02issue09-02.

[24] S. R. Gudi, "Enhancing Optical Character Recognition (OCR) Accuracy in Healthcare Prescription Processing using Artificial Neural Networks," European Journal of Artificial Intelligence and Machine Learning, vol. 4, no. 6, pp. 1–6, Nov. 2025, doi: https://doi.org/10.24018/ejai.2025.4.6.79.

[25] A. K. K. Varma Alluri, "Salesforce CRM Framework for Real Time DeFi Portfolio Intelligence and Customer Engagement Forecasting in Web3 Based Decentralized Finance Ecosystems Using ML Techniques," International Journal of AI, BigData, Computational and Management Studies, vol. 6, 2025, doi: https://doi.org/10.63282/3050-9416.ijaibdcms-v6i4p111.

[26] Shrutika Prakash Mokashi, Prahlad Chowdhury, and Guru Lakshmi Priyanka Bodagala, "Smart Manufacturing and the Operator's Digital Double: Modeling Cognitive Load Through a Psychosocial Digital Twin," International Journal of Sustainability and Innovation in Engineering, vol. 4, no. 1, Mar. 2026, doi: https://doi.org/10.56830/ijsie202602.

[27] S. R. Gudi, "Ensuring Secure and Compliant Fax Communication: Anomaly Detection and Encryption Strategies for Data in Transit," 2025 4th International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), pp. 786–791, Sep. 2025, doi: https://doi.org/10.1109/icimia67127.2025.11200537.

[28] S. K. Gunda, "Comparative Analysis of Machine Learning Models for Software Defect Prediction," pp. 1–6, Oct. 2024, doi: https://doi.org/10.1109/icpects62210.2024.10780167

[29] P. Chowdhury, "Human-Robot Collaboration (HRC) in Automotive: SAP DM Orchestration of Cobot Work-Cells," American Journal of Technology, vol. 4, no. 4, pp. 87–100, Dec. 2025, doi: https://doi.org/10.58425/ajt.v4i4.466.

[30] S. D. Sivva, "An End-to-End AI-Based Systems Engineering Paradigm for Lifecycle Governance, Predictive Quality Assurance, Automation Economics, and Cybersecurity Intelligence," Journal of Frontiers in Multidisciplinary Research, vol. 4, no. 1, pp. 600–604, 2023, doi: https://doi.org/10.54660/.jfmr.2023.4.1.600-604.

[31] S. R. Gudi, "Leveraging Predictive Analytics and Redis-Backed Caching to Optimize Specialty Medication Fulfillment and Pharmacy Inventory Management," International Journal of AI, BigData, Computational and Management Studies, vol. 5, no. 3, Oct. 2024, doi: https://doi.org/10.63282/3050-9416.ijaibdcms-v5i3p116.

[32] P. Chowdhury, "A Cloud-Native Decision Intelligence Architecture for Sustainable CPG Supply Chain Networks," Journal of Engineering Research and Sciences, vol. 5, no. 1, p. 35, Jan. 2026, doi: https://doi.org/10.55708/js0501004.

[33] R. R. Thalakanti, "Optimizing Neural Network Architecture for Binary Classification Using Evolutionary Algorithms," 2025 International Conference on Electronics and Computing, Communication Networking Automation Technologies (ICEC2NT), pp. 1–6, Sep. 2025, doi: https://doi.org/10.1109/icec2nt65402.2025.11380048.

[34] Sai Krishna Gunda, "An exploration of adaptive ensemble approaches in software fault detection: Balancing accuracy and robustness," THE FIRST INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ARTIFICIAL INTELLIGENCE, CYBER SECURITY, AND EMBEDDED SYSTEMS: ICRTACES2024, vol. 3345, no. 1, 7 January 2026, Doi: https://doi.org/10.1063/5.0298093

[35] Srikanth Reddy Gudi, "A Comparative Analysis of Pivotal Cloud Foundry and OpenShift Cloud Platforms," The American Journal of Applied Sciences, vol. 7, no. 7, pp. 20-29, 2025, doi: https://doi.org/10.37547/tajas/Volume07Issue07-03

[36] R. R. Thalakanti, "Enhancing Convergence in Fully Connected Neural Networks via Optimized Backpropagation," 2025 2nd International Conference on Computing and Data Science (ICCDS), pp. 1–6, Jul. 2025, doi: https://doi.org/10.1109/iccds64403.2025.11209625.

[37] P. Chowdhury, "Global MES Rollout Strategies: Overcoming Localization Challenges in Multi-Country Deployments," The American Journal of Applied Sciences, vol. 7, no. 07, pp. 30–28, Jul. 2025, doi: https://doi.org/10.37547/tajas/volume07issue07-04.

[38] R. R. Thalakanti, "Convergence Analysis and Implementation of Linear Multistep Methods for Solving Ordinary Differential Equations," 2025 2nd Asian Conference on Intelligent Technologies (ACOIT), pp. 1–18, Oct. 2025, doi: https://doi.org/10.1109/acoit66109.2025.11436783.

[39] Sai Krishna Gunda, "Advancing software fault detection: A comparative study of neural network architectures," THE FIRST INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ARTIFICIAL INTELLIGENCE, CYBER SECURITY, AND EMBEDDED SYSTEMS: ICRTACES2024, vol. 3345, no. 1, 7 January 2026, doi: https://doi.org/10.1063/5.0298095

[40] S. K. Gunda, "Automatic Software Vulnerabilty Detection Using Code Metrics and Feature Extraction," 2025 2nd International Conference On Multidisciplinary Research and Innovations in Engineering (MRIE), pp. 115–120, Jul. 2025, doi: https://doi.org/10.1109/mrie66930.2025.11156601.