International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

From Reactive ITSM to Predictive Incident Prevention Using AI

Nayan Kumar Sureshbhai Patel

Abstract: Traditional IT service management (ITSM) relies on a reactive operating model in which operations teams respond to failures after end users are already affected. This approach is increasingly insufficient in cloud-native, distributed, and always-on enterprise environments where the cost of delayed incident response includes downtime, reputational damage, and compounding operational waste. This article examines how artificial intelligence transforms ITSM from reactive response to predictive incident prevention through four core capabilities: anomaly detection, failure prediction, risk scoring, and proactive automation. Drawing on a structured review of 22 sources spanning AIOps, machine learning operations, enterprise integration, and sustainable computing, the article proposes a unified framework that connects AI-driven prevention capabilities to the six architectural patterns, microservices decomposition, event-driven integration, containerization, orchestration, caching, and observability, that are required to sustain them in production. The article presents three tables and three figures that structure the reactive-to-predictive transition as a maturity model, detail the AI capability stack, and map architectural contributions to sustainability outcomes. Findings indicate that predictive ITSM delivers measurable resilience improvements and sustainability co-benefits when implemented within an architecturally disciplined platform. The article concludes with governance requirements for automated decision-making and directions for future empirical research.

Keywords: *AIOps, anomaly detection, enterprise resilience, IT service management, predictive incident prevention*

1. Introduction

Enterprise IT environments have grown dramatically in scale and complexity over the past decade. Cloud-native architecture, microservices-based application architecture, containerized workloads and multi-cloud deployments can expose operations teams to a large number of independent services, dependencies and failure modes. IT Service Management frameworks, such as those based on ITIL principles, can help to manage this situation by providing a formalized set of standard processes to detect, classify, route and resolve incidents. The result is a reactive operating model: teams respond to failures after users are affected, measure success by how quickly service is restored, and improve by analyzing incidents after the fact.

The reactive model has significant limitations in always-on environments. When failure is the trigger for response, the time between a developing problem and the moment it becomes operationally visible is entirely wasted opportunity for prevention. In distributed systems with thousands of interdependent services, failures rarely appear without antecedent signals: latency trends, error rate escalations, resource exhaustion patterns, and dependency topology changes all precede major incidents. The AIOps literature has documented these patterns systematically. For failure management, Notaro et al. describe machine learning techniques for anomaly detection, event correlation, and root cause analysis. They demonstrated the technical feasibility of predictive failure prevention in a wide range of enterprise scenarios [16]. Islam et al.'s recent industry study using IBM Cloud telemetry reinforces these findings at operational scale, demonstrating that ML-based anomaly detection across hundreds of cloud components substantially reduces the DevOps team's manual monitoring burden and decreases the risk of service outages [24]. According to Dang et al. the foremost barrier for enterprises to adopt AIOps at scale is the gap between technical feasibility and operational reality [5].

Sculley et al.'s hidden technical debt framework adds a critical constraint: ML systems deployed in ITSM contexts accumulate operational debt in the form of unstable data dependencies, undeclared consumers, feedback loops, and configuration complexity that degrades predictive accuracy over time if the surrounding platform is not

Independent Researcher, USA
ORCID: 0009-0003-0175-5704

architecturally prepared to sustain them [18]. This constraint explains why organizations that add AI detection to architecturally unprepared platforms frequently experience the same reactive patterns they intended to replace: the AI layer produces alerts that operations teams cannot act on quickly enough, generates false positives that erode trust, or fails silently when upstream data sources change without warning.

The transformation to predictive ITSM therefore requires both AI capability and architectural discipline. Figure 1 illustrates the four-layer platform architecture through which predictive prevention is operationalized: a data foundation that provides clean, lineage-tracked telemetry; an intelligence layer that applies AI to surface early warning signals; a workflow layer that converts those signals into governed automated actions; and an observability and governance layer that makes AI behavior auditable and correctable. The four AI capabilities that define the predictive operating model, anomaly detection, failure prediction, risk scoring, and proactive automation, function across all four layers, as producers and consumers of structured information.

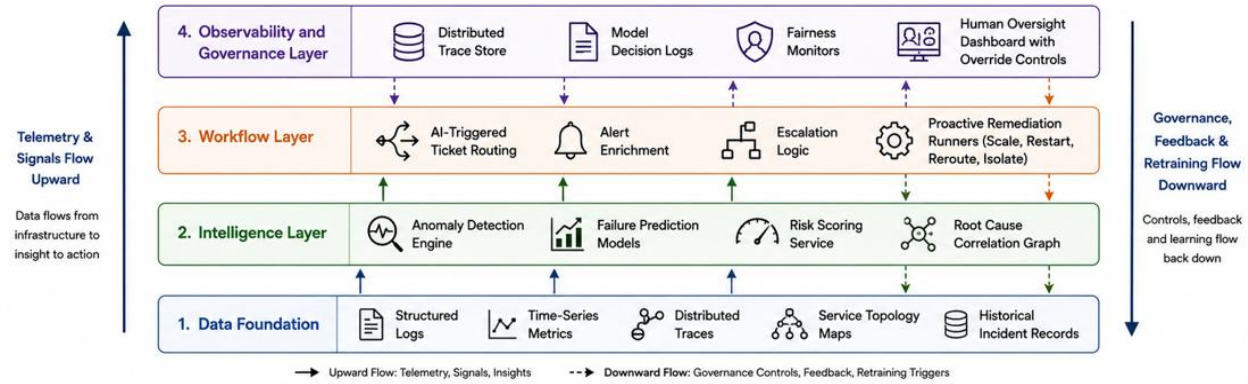

Fig. 1. Four-layer predictive ITSM platform architecture showing telemetry-to-action signal flow and governance feedback loop.

This article makes three contributions. First, it synthesizes the AIOps and platform engineering literatures into a unified predictive ITSM framework organized around the four-layer architecture in Figure 1. Second, it identifies the six architectural patterns, microservices decomposition, event-driven integration, containerization, orchestration, caching, and observability, that are prerequisites for sustaining predictive prevention at an enterprise scale. Third, it examines sustainability as a structural co-benefit of architecturally disciplined predictive operations, demonstrating that each of the six patterns reduces computational waste independently and that their combined effect compounds across the platform lifecycle.

2. Method

This article employs a structured literature review methodology to synthesize current knowledge on AI-driven predictive ITSM and the architectural patterns required to support it. The method has three stages: identification of sources, selection of sources, and thematic synthesis.

Accompanying studies were found in the ACM Digital Library, IEEE Xplore, and open-access archives hosted by USENIX by searching for AIOps, predictive incident management, IT service management automation, anomaly detection in production systems, machine learning operations, site reliability engineering, enterprise integration patterns, containerization and orchestration and sustainable AI computing. Related works were also identified when mentioned in primary sources, including technical O'Reilly books on Site Reliability Engineering [2] and Data-Intensive Applications [10], and influential professional literature [12].

Selection criteria required that the publication was research, formally published or from an established professional publisher, in English, published between 2003 and 2023, and describes one or more of the following: AIOps and incident management, enterprise architecture and integration, infrastructure and operations, ML systems and operations, observability and governance, and sustainability. The earlier date (2003) was selected as it includes Hohpe and Woolf's reference [8]

which described enterprise integration patterns. The selection process yielded 22 primary sources. These were analyzed for contribution to predictive prevention capability, architectural requirement, governance implication, and sustainability relevance. The synthesis is organized around the unified framework in Figure 1, with Tables 1 through 3 and Figures 2 and 3 developed from the thematic analysis.

3. Results and Discussion

3.1 Core AI Capabilities for Predictive Incident Prevention

The shift from reactive to predictive ITSM is defined by four AI capabilities that together create a continuous prevention cycle. Table 1 contrasts the reactive and predictive operating models across nine key dimensions, establishing the operational context. Table 2 details the mechanism, input signals, operational outcomes, and citations for each capability.

Table 1. Reactive versus predictive ITSM: comparison across nine operational dimensions.

Dimension	Reactive ITSM	Predictive ITSM
Trigger	User-reported failure or alert threshold breach	AI-detected early warning signal or anomaly score
Detection time	After degradation is user-visible	Before impact reaches end users
Primary data source	Incident tickets and monitoring thresholds	Telemetry streams, logs, topology maps, historical patterns
Response mechanism	Manual triage and escalation	Proactive automation with human-in-the-loop oversight
AI involvement	Limited; rule-based alerting	Core: anomaly detection, failure prediction, risk scoring
Failure model	Reactive repair after disruption	Anticipatory prevention through continuous risk assessment
Sustainability posture	High compute waste from repeated incident cycles	Reduced waste through targeted, event-driven responses
Governance model	Post-incident review and audit	Continuous observability with real-time governance controls
Organizational outcome	Recovery speed (MTTR focus)	Prevention rate and resilience (MTTD + prevention focus)

Table 2. Core AI capabilities for predictive incident prevention: mechanisms, inputs, outcomes, and citations.

AI Capability	Mechanism	Input Signals	Operational Outcome	Key Citation
Anomaly Detection	Statistical and ML-based deviation from learned baseline	Metrics, logs, traces, latency patterns	Early surfacing of abnormal behavior before escalation	[6, 7, 16]
Failure Prediction	Time-series forecasting; pattern matching against historical failures	Error rate trends, resource exhaustion signals, topology changes	Pre-incident warning enabling preventive action	[4, 13, 19]
Risk Scoring	Probabilistic ranking of service health; blast-radius estimation	Service dependencies, SLA tiers, recent anomaly history	Prioritized attention for operations teams	[5, 16]
Proactive Automation	Policy-driven execution triggered by AI-generated risk assessments	Risk score thresholds, runbook mappings, orchestration APIs	Automated remediation: scale, restart, reroute, isolate	[11, 21, 22]
Root Cause Analysis	Graph traversal of service dependencies; decision tree diagnosis	Service topology maps, correlated event timelines	Faster isolation of fault origin; reduced investigation time	[4, 19]

Anomaly detection is the first line of predictive prevention. Rather than waiting for a monitored metric to cross a static threshold, ML-based anomaly detectors learn the normal behavioral profile of each service and surface deviations that exceed expected variance. This is fundamentally different from threshold-based alerting, which misses deviations that are individually below threshold but collectively indicative of developing failure. He et al.'s Drain algorithm addresses one of the most practical challenges in log-based anomaly detection: parsing the high-volume, semi-structured log streams that enterprise services generate in real time [6]. Drain uses a fixed-depth parse tree to extract structured event templates from unstructured log text online, without requiring a predefined schema, significantly reducing detection latency. He et al.'s survey of performance bugs complements this by documenting the behavioral signatures, memory leak patterns, lock contention profiles, and I/O saturation trends, that reliably precede specific categories of production failure [7]. Notaro et al.'s AIOps survey catalogs the full range of applicable ML approaches, from unsupervised clustering over event sequences to supervised classification of labeled incident histories [16].

Failure prediction extends detection from identifying current anomalies to forecasting future failure states. An anomaly detector signals that something unusual is happening now; a failure predictor forecasts what is likely to happen next and estimates the preventive window available. Chen et al.'s decision tree-based failure diagnosis established that historical incident records can train classifiers to identify which combinations of observed signals have historically preceded specific failure categories [4]. Li et al.'s automated incident detection system for large-scale cloud environments demonstrates that deep learning approaches can forecast incidents with sufficient lead time, on the order of minutes to tens of minutes ahead of user-visible impact, to enable preventive action in production [13]. Soldani and Brogi's survey combines failure prediction with blast-radius estimation, mapping the likely propagation path of a predicted failure through the service dependency graph to identify which downstream services are at risk [19]. Zhang et al.'s 2025 comprehensive survey of 98 microservice failure diagnosis studies advances this landscape further, cataloguing how cascading failures in dynamically interacting services demand both detection and predictive diagnosis capabilities that extend well beyond static threshold alerting [25].

Risk scoring translates anomaly signals and failure predictions into prioritized operational intelligence. In a large enterprise, multiple warnings may be active simultaneously. Otherwise, the operations team is left to process an undifferentiated "all alerts" queue with no way of knowing which ones to prioritize. The risk scoring layer computes the probability and potential impact of each active signal to the business. The signals are ranked by their operational priority based on AI risk vulnerabilities, business context, service criticality tier, SLA commitments, and impact to active customers. Risk scoring translates anomaly signals into prioritized operational intelligence. In large enterprise environments, multiple concurrent warnings make undifferentiated alert queues operationally unmanageable. Dang et al. identify prioritization as one of the most practically valuable features of AIOps platforms [5]; Yu et al.'s comprehensive survey of intelligent alert and incident management systems extends this finding, documenting the full architecture from alert representation to incident resolution and demonstrating that automated triage reduces both MTTR and engineer escalation load in production deployments [23].

Proactive automation closes the prevention loop by prioritizing risk and taking action. The automation engine is responsible for executing remediation playbooks that have been pre-authorized/approved by corresponding teams when risk scores exceed configurable thresholds. Such remediation actions would include scaling up workloads with memory pressure; restarting workloads reaching unhealthy states; re-routing workloads away from underperforming nodes; and isolating workloads that produce abnormal output. Kreuzberger et al.'s MLOps framework explains how the automation of models is integrated into CD pipelines to create automated low-latency, versioned responses [11]. Syed et al.'s framework describes robotic process automation (RPA) that is triggered by AI to execute specific remediations based on AI-managed detection, comprehension, validation, and prioritization [21]. Zaharia et al.'s MLflow model provides the lifecycle infrastructure, experiment tracking, model registry, and deployment tooling for managing the models that power automated prevention at production scale [22].

3.2 Architectural Foundations for Predictive ITSM

The four AI capabilities described in Section 3.1 require an architectural foundation that provides signal connectivity, modularity, and operational discipline. Figure 2 frames this requirement as a four-stage maturity progression. Table 3 details the contribution of each architectural pattern to predictive ITSM outcomes and sustainability.

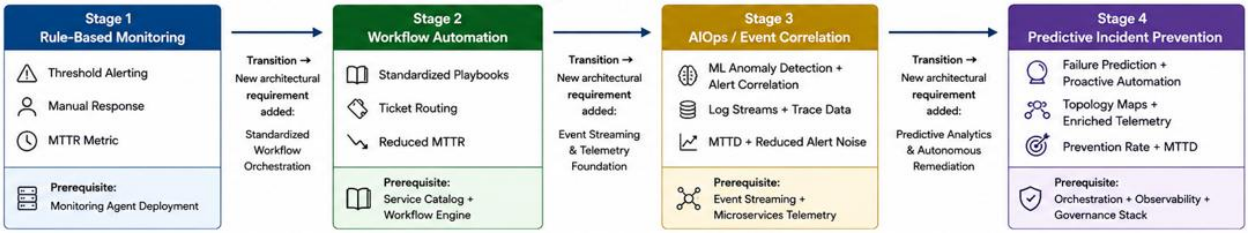

Fig. 2. Four-stage maturity model from rule-based monitoring to predictive incident prevention, annotated with architectural prerequisites at each transition.

Table 3. Architectural patterns: predictive ITSM contributions and sustainability benefits.

Architectural Pattern	Predictive ITSM Contribution	Sustainability Benefit	Key Citation
Microservices decomposition	Granular service-level observability; fault isolation; independent scaling of detection services	Right-sized resource allocation eliminates monolith overprovisioning	[12, 18]
Event-driven integration	Real-time signal propagation; activates AI logic only on meaningful state changes	Eliminates polling waste; reduces idle compute across the platform	[8, 10]
Containerization (Docker)	Consistent deployment of anomaly detection and automation services across environments	Higher resource density; reduced per-service infrastructure overhead	[15, 18]
Orchestration (Kubernetes)	Dynamic scheduling of AI workloads; auto-scaling detection pipelines; self-healing services	Adaptive resource use; off-peak batch scheduling; energy-aware placement	[3, 11]
Caching layers	Stores topology maps, enriched alerts, inference results for rapid reuse	Reduces redundant compute for repeated operational queries	[17, 20]
Observability stack	Distributed tracing, structured logging, and metrics enable full AI decision auditability	Early detection of efficiency regressions prevents waste from compounding	[2, 6, 13]

Microservices decomposition is the first architectural prerequisite for effective service-level AI detection [12]. A monolithic application presents anomaly detection as a signal-averaging problem: the aggregate behavior of many functions blended together makes it difficult to isolate which component exhibits abnormal behavior. Microservices make each service independently observable, with its health metrics, structured logs, and distributed trace participation. This granularity enables failure prediction models to be trained per-service or per-service-group, significantly improving prediction precision compared to models trained on blended system-wide telemetry. Lewis and Fowler identify independent deployability and organizational alignment as the defining characteristics of microservices [12]; in a predictive ITSM context, independent observability is equally foundational. Failure isolation improves automated response precision as well: when one service begins to fail, AI systems can correlate the event with its dependency map, estimate blast radius, and trigger a targeted remediation without broad, disruptive system-wide intervention.

Event-driven integration is the signal infrastructure that activates AI processing in response to meaningful operational state changes rather than on fixed polling schedules [8, 10]. Chaudhari's analysis of event-driven architecture in enterprise systems corroborates this sustainability argument, documenting how the elimination of polling-based integration and its replacement with reactive, event-triggered processing reduces idle compute cycles and enables right-sized resource consumption across

distributed service platforms [28]. Hohpe and Woolf's enterprise integration patterns establish the message channel, publish-subscribe, and content-based routing primitives that allow operational events, a latency spike, a queue depth threshold, a configuration change, and a security anomaly to propagate immediately to the AI services that need to act on them [8]. Kleppmann's event-log model extends the concept to the continuous-stream paradigm in which enterprise state is recorded as an ordered, append-only log that AI consumers can read in real time or replay historically [10]. Thalary and Katipelly extend this foundation to contemporary cloud-native practice, demonstrating how event streaming, Infrastructure as Code, and Kubernetes-native observability work in concert to reduce operational latency and improve fault isolation across distributed service topologies [26]. For predictive ITSM, event streaming provides two critical capabilities unavailable in polling-based integration: historical replay enables prediction models to be backtested against the actual event sequence that preceded previous incidents; and real-time consumption enables anomaly detection to operate at the latency of the originating event rather than the polling interval.

3.3 Infrastructure Optimization

Architectural patterns define the logical structure of predictive ITSM; infrastructure optimization determines whether that structure performs efficiently at an enterprise scale. Containerization with Docker provides the deployment consistency that MLOps requires [15]. The anomaly detection model that performs correctly in development must behave identically in staging and production, with the same library versions, numerical precision settings, and system resource constraints. Without this consistency, model behavior drifts between environments in ways that produce systematic prediction errors. Merkel's introduction of Docker established that containers achieve this consistency with substantially lower per-service overhead than virtual machine-based deployment, sharing the host OS kernel rather than virtualizing it entirely [15]. In a predictive ITSM platform with many concurrently running AI services, anomaly detectors, failure predictors, risk scorers, automation runners, and observability agents, container-based deployment improves hardware utilization by allowing more services to coexist on shared infrastructure at higher density.

Kubernetes orchestration converts containerized infrastructure into a self-managing, adaptive workload management layer [3]. Burns et al.'s documentation of the Kubernetes design lineage establishes the key principle: workload requirements expressed as desired state are continuously reconciled with actual state by the orchestration system, without manual intervention [3]. Arafat et al.'s benchmarking study of next-generation event-driven architectures deployed on Kubernetes confirms this principle empirically, showing that intelligent orchestration of messaging frameworks reduces latency and improves scalability under production-realistic workloads compared to statically configured alternatives [27]. For predictive ITSM, anomaly detection pipelines can scale horizontally when telemetry volume spikes during an infrastructure disruption; inference services restart automatically if they fail health checks; and batch model-retraining jobs run during off-peak hours to minimize competition with production detection workloads. Kreuzberger et al. identify orchestration-integrated deployment as one of the MLOps maturity indicators that distinguishes operationally sustainable ML platforms from those that degrade over time [11].

Caching reduces the computational cost of the repetitive information lookups that predictive ITSM generates continuously: service topology maps for dependency evaluation, SLA tier definitions for risk scoring, recent incident histories for pattern matching, and enriched alert context for automation playbook selection. Re-fetching all this information from primary data stores on every request creates high database load, elevated latency, and unnecessarily high compute consumption. Caching these in memory or fast-access storage layers addresses all three problems simultaneously. Schwartz et al.'s Green AI framework positions computation efficiency as a first-class operational objective: practitioners should select the minimum model capable of meeting detection requirements and design data pipelines that avoid unnecessary recomputation [17]. Strubell et al.'s energy accounting for deep learning provides an empirical foundation, demonstrating that inference cost scales with model complexity in ways that compound significantly at production request volumes [20].

3.4 Observability, Governance, and Sustainability

Predictive incident prevention is only as operationally valuable as the organization's ability to

understand, verify, and correct AI behavior in production. Figure 3 maps the sustainability impact of the six architectural patterns across three dimensions: compute waste reduction, energy efficiency, and infrastructure density.

Architectural Pattern	Compute Waste Reduction (Impact & Justification)	Energy Efficiency (Impact & Justification)	Infrastructure Density (Impact & Justification)
1. Microservices Decomposition	**High** Eliminates monolith-wide scaling for localized issues.	**Medium** Smaller services consume less when right-sized.	**Medium** Finer granularity improves packing and resource allocation.
2. Event-Driven Integration	**High** Activates processing only on meaningful state changes.	**Medium** Reduces unnecessary polling and constant checks.	**Medium** Asynchronous decoupling improves resource sharing.
3. Containerization	**Medium** Improves resource isolation and right-sizing.	**Medium** Lighter runtime reduces overhead vs. VMs.	**High** Shared kernel enables multiple services per host.
4. Kubernetes Orchestration	**High** Auto-scaling and bin-packing minimize idle capacity.	**High** Off-peak scheduling and adaptive scaling reduce idle consumption.	**High** Dense packing and node utilization maximize infrastructure use.
5. Caching Layers	**High** Eliminates redundant computation for repeated queries.	**Medium** Fewer backend calls lower energy per transaction.	**Medium** Reduces load, enabling higher consolidation ratios.
6. Observability Stack	**Medium** Faster issue detection prevents wasteful resource usage.	**Medium** Early detection of efficiency regressions prevents compounding waste.	**Medium** Better visibility improves capacity planning and placement.

Impact Rating: High — Significant positive impact on usstainability outcome; Medium — Moderate positive impact on sustainability outcome; Low — Minimal or limited positive impact on sustainability outcome

Fig. 3. Sustainability impact matrix: six architectural patterns mapped to compute waste reduction, energy efficiency, and infrastructure density.

Beyer et al.'s framework for site reliability engineering defines the observability stack as the combination of distributed traces, structured logs, and time-series metrics that allows the internal state of a production system to be inferred from its external outputs [2]. For predictive ITSM, this stack must be extended to include AI-specific telemetry: the model version that produced each risk score, the feature values that drove that score, the confidence interval of each prediction, and the complete chain of automated actions triggered by each prediction event. Without this extended observability, the AI layer operates as a black box, capable of generating actions and affecting business outcomes, but incapable of supporting the audit, debugging, continuous improvement, and governance review cycles that responsible AI deployment requires. He et al.'s Drain algorithm contributes to this observability infrastructure by enabling real-time structured extraction from the high-volume unstructured log streams that both the operational environment and the AI services themselves generate [6].

Governance requirements for automated decision-making in ITSM contexts are substantial and increasing. Jobin et al.'s synthesis of 84 global AI ethics frameworks identifies transparency, accountability, fairness, and human oversight as the most consistently endorsed principles across governmental, intergovernmental, and corporate guidance documents [9]. Translated to predictive ITSM, these require that every automated remediation action be logged with its triggering evidence and the identity of the model version that generated it; that risk scores be auditable by designated reviewers; that automated decisions affecting service availability for specific user populations be tested for differential impact; and that human override capability be present at every automation decision point. Adadi and Berrada's XAI survey provides the technical means through which these requirements can be met [1]: explainability methods, local feature attribution, counterfactual explanations, and decision boundary visualization, translate model outputs into human-readable justifications that satisfy audit requirements for compliance officers, operations engineers, and business stakeholders. Mehrabi et al.'s fairness survey addresses the bias dimension: AI models trained on historical incident data may perpetuate historical patterns of differential service reliability if not tested and monitored for bias continuously [14].

The sustainability contribution of predictive ITSM is structural and compounding. Each of the six architectural patterns in Table 3 reduces computational waste independently: microservices eliminate monolith-wide scaling for localized issues; event-driven design activates processing only on meaningful state changes; containerization improves hardware utilization density; orchestration enables energy-aware workload scheduling; caching eliminates redundant computation; and observability enables early detection of efficiency regressions before they compound. Schwartz et al.'s Green AI position finds its concrete expression in these six patterns [17]: together they create a platform that is measurably more resource-efficient than the reactive ITSM environment it replaces, converting architectural discipline into operational sustainability.

4. Conclusion

This article has presented a unified framework for AI-driven predictive incident prevention in enterprise ITSM environments, connecting four core AI capabilities, anomaly detection, failure prediction, risk scoring, and proactive automation, to the six architectural patterns required to sustain them in production: microservices decomposition, event-driven integration, containerization, Kubernetes orchestration, caching, and observability with governance. The central finding is that predictive ITSM is an architectural discipline, not a monitoring upgrade. Organizations that treat AI-driven prevention as a layer added on top of existing reactive infrastructure encounter the same hidden technical debt, data dependency failures, and governance gaps that limit reactive ITSM at scale. The framework in Figure 1 and the maturity model in Figure 2 provide practical tools for assessing current state and identifying the architectural investments required for predictive capability. The sustainability argument in Figure 3 and Table 3 makes the case that architectural investment in predictive ITSM delivers compound returns: organizational resilience and resource efficiency gained simultaneously from the same design commitments. Future research priorities include empirical measurement of predictive prevention rates against reactive baselines in real enterprise deployments; governance framework development for automated remediation in regulated industries; multi-cloud observability architectures that maintain AI prediction accuracy across heterogeneous infrastructure providers; and lifecycle energy accounting models that quantify the sustainability impact of predictive versus reactive ITSM at enterprise scale.

Acknowledgments

The author thanks the reviewers and editorial team of the International Journal of Intelligent Systems and Applications in Engineering for their consideration of this manuscript.

Author Contributions

Nayan Kumar Sureshbhai Patel: Conceptualization, Methodology, Writing, Reviewing, and Editing Original Draft.

Conflicts of Interest

The author declares no conflict of interest.

References

[1] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," IEEE Access, vol. 6, pp. 52138–52160, 2018. Available: https://doi.org/10.1109/ACCESS.2018.2870052

[2] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems, O'Reilly Media, Sebastopol, CA, 2016. Available: https://repo.darmajaya.ac.id/4636/1/Site%20Reliability%20Engineering_%20How%20Google%20Runs%20Production%20Systems%20%28%20PDFDrive%20%29.pdf

[3] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," Communications of the ACM, vol. 59, no. 5, pp. 50–57, 2016. Available: https://dl.acm.org/doi/fullHtml/10.1145/2890784

[4] M. Chen, X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in Proc. IEEE International Conference on Autonomic Computing (ICAC), New York, 2004, pp. 36–43. Available: https://doi.org/10.1109/ICAC.2004.1301345

[5] Y. Dang, Q. Lin, and P. Huang, "AIOps: Real-world challenges and research innovations," in Proc. 41st IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Montreal, 2019, pp. 4–5. Available: https://doi.org/10.1109/ICSE-Companion.2019.00023

[6] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in Proc. IEEE International Conference on Web Services (ICWS), Honolulu, 2017, pp. 33–40. Available: https://doi.org/10.1109/ICWS.2017.13

[7] S. He, J. Zhu, P. He, and M. R. Lyu, "Survey on real-world performance bugs," ACM Computing Surveys, vol. 54, no. 3, pp. 1–35, 2021. Available: https://dl.acm.org/doi/abs/10.1145/3460345

[8] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Professional, Boston, MA, 2003. Available: https://dl.acm.org/doi/book/10.5555/940308

[9] A. Jobin, M. Ienca, and E. Vayena, "The global landscape of AI ethics guidelines," Nature Machine Intelligence, vol. 1, pp. 389–399, 2019. Available: https://www.nature.com/articles/s42256-019-0088-2

[10] M. Kleppmann, Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media, Sebastopol, CA, 2017. Available: https://www.oreilly.com/library/view/designing-data-intensive-applications/9781491903063/

[11] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (MLOps): Overview, definition, and architecture," IEEE Access, vol. 11, pp. 31866–31879, 2023. Available: https://doi.org/10.1109/ACCESS.2023.3262138

[12] J. Lewis and M. Fowler, "Microservices: A definition of this new architectural term," MartinFowler.com, 2014. Available: https://martinfowler.com/articles/microservices.html

[13] L. Li, X. Zhang, X. Zhao, H. Zhang, Y. Kang, P. Zhao, B. Qiao et al., "Fighting the fog of war: Automated incident detection for cloud systems," in Proc. USENIX Annual Technical Conference (USENIX ATC 21), 2021, pp. 131–146. Available: https://www.usenix.org/conference/atc21/presentation/li-liqun

[14] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," ACM Computing Surveys, vol. 54, no. 6, pp. 1–35, 2021. Available: https://dl.acm.org/doi/abs/10.1145/3457607

[15] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," Linux Journal, vol. 2014, no. 239, pp. 2, 2014. Available: https://www.seltzer.com/margo/teaching/CS508.19/papers/merkel14.pdf

[16] P. Notaro, J. Cardoso, and M. Gerndt, "A survey of AIOps methods for failure management," ACM Transactions on Intelligent Systems and Technology, vol. 12, no. 6, pp. 1–45, 2021. Available: https://dl.acm.org/doi/full/10.1145/3483424

[17] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," Communications of the ACM, vol. 63, no. 12, pp. 54–63, 2020. Available: https://dl.acm.org/doi/10.1145/3381831

[18] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," Advances in Neural Information Processing Systems, vol. 28, 2015. Available: https://proceedings.neurips.cc/paper_files/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html

[19] J. Soldani and A. Brogi, "Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey," ACM Computing Surveys, vol. 55, no. 3, pp. 1–39, 2022. Available: https://dl.acm.org/doi/full/10.1145/3501297

[20] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in Proc. 57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, 2019, pp. 3645–3650. Available: https://aclanthology.org/P19-1355/

[21] R. Syed, S. Suriadi, M. Adams, W. Bandara, S. J. J. Leemans, C. Ouyang, A. H. M. ter Hofstede, I. van de Weerd, M. T. Wynn, and H. A. Reijers, "Robotic process automation: Contemporary themes and challenges," Computers in Industry, vol. 115, pp. 103162, 2020. Available: https://doi.org/10.1016/j.compind.2019.103162

[22] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, F. Xie, and C. Zumar, "Accelerating the machine learning lifecycle with MLflow," IEEE Data Engineering Bulletin, vol. 41, no. 4, pp. 39–45, 2018. Available: http://sites.computer.org/debull/A18dec/A18DEC-CD.pdf#page=41

[23] Q. Yu, N. Zhao, M. Li, Z. Li, H. Wang, W. Zhang, K. Sui and D. Pei, "A survey on intelligent management of alerts and incidents in IT services," Journal of Network and Computer Applications, vol.

224, 2024. Available: https://doi.org/10.1016/j.jnca.2024.103842
[24] Mohammad Saiful Islam, Mohamed Sami Rakha, William Pourmajidi, Janakan Sivaloganathan, John Steinbacher, and Andriy Miranskyy, "Anomaly detection in large-scale cloud systems: An industry case and dataset," in Proc. 47th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Ottawa, Canada, 2025, pp. 377–388. Available: https://doi.org/10.1109/ICSE-SEIP66354.2025.00039
[25] Shenglin Zhang, Sibo Xia, Wenzhao Fan, Binpeng Shi, Xiao Xiong, Zhenyu Zhong, Minghua Ma, Yongqian Sun, and Dan Pei, "Failure diagnosis in microservice systems: A comprehensive survey and analysis," ACM Transactions on Software Engineering and Methodology, vol. 35, no. 1, pp. 1–55, 2025. Available: https://dl.acm.org/doi/10.1145/3715005
[26] Sumith Thalary and Anvesh Katipelly, "Cloud-native design for event-driven systems: Where software architecture decisions meet DevOps reality," International Journal of AI, BigData, Computational and Management Studies, vol. 5, no. 2, pp. 202–211, June 2024. Available: http://ijaibdcms.org/index.php/ijaibdcms/article/view/509
[27] Juneyd Arafat, Faria Tasmin, Saurabh Poudel, and Abdullah Tareq, "Next-generation event-driven architectures: Performance, scalability, and intelligent orchestration across messaging frameworks," arXiv preprint arXiv:2510.04404, October 2025. Available: https://arxiv.org/abs/2510.04404
[28] Sagar Chaudhari, "Event-driven architecture: Building responsive enterprise systems," International Journal of Scientific Research in Computer Science, Engineering and Information Technology, vol. 11, no. 1, pp. 3063–3073, February 2025. Available: https://doi.org/10.32628/CSEIT251112323