Green Intelligent Systems and Applications

Research Article
Volume 6(1), 2026, 1–18
https://doi.org/10.53623/gisa.v6i1.963

Integrating TOGAF 10 and ISO 20000-1:2018 for Digital Multi-finance Service Level Agreement/Mean Time to Repair improvements

Bagus Resa Destian*, Panca Dewi Pamungkasari

Universitas Nasional Faculty of Communication and Information Technology, Jakarta, Indonesia

*Correspondence: bagusresadestian.2024@civitas.unas.ac.id

SUBMITTED: 13 December 2025; REVISED: 29 January 2026; ACCEPTED: 2 February 2026

ABSTRACT: Digital transformation in the multi-finance sector demands service architectures that are flexible, reliable, and scalable; however, misalignment between architectural design and operational execution often leads to weak service performance. This study proposes an integrated framework that combines TOGAF 10 artifacts with ISO/IEC 20000-1:2018 processes to systematically estimate Service Level Agreement (SLA) targets and reduce Mean Time to Repair (MTTR). Using a Design Science Research approach, the framework was implemented in a 14-month case study at PT XYZ Multi-finance. The resulting artifacts include a bidirectional traceability model linking business objectives to SLA and MTTR indicators, as well as an operability pattern catalog to support "design for operability." The implementation delivered measurable operational improvements: MTTR decreased from a peak of 775 minutes to below 60 minutes, Mean Time to Detect (MTTD) was reduced by approximately 90%, SLA compliance increased to 99.7%, and incidents caused by manual configuration errors declined. These results demonstrate that integrating enterprise architecture design with service management processes can significantly improve service reliability and overall operational performance.

KEYWORDS: TOGAF 10; ISO/IEC 20000-1:2018; service level agreement; design science research; site reliability engineering

1. Introduction

Digital transformation in the multi-finance sector requires service operations that are resilient, predictable, and compliant with increasingly strict regulations. As financial services become more dependent on digital platforms, organizational performance is shaped by structured operational governance supported by indicators such as Service Level Agreements (SLA) and MTTR. Although the importance of aligning Enterprise Architecture (EA) with Information Technology Service Management (ITSM) has been recognized, many organizations still experience a persistent gap between design and operations, where strategic architecture plans are not translated into measurable operational results. Existing studies indicate that although frameworks such as TOGAF provide comprehensive architectural guidance, their operational relationships remain largely conceptual and do not explicitly

address service reliability or incident recovery performance [1]. Other research explains that while digital infrastructure has become an important component of business continuity, mechanisms that formally link architectural decisions with operability such as configuration integrity, recovery capabilities, and reliability targets, are often missing [2, 3]. The emergence of approaches such as BizDevOps has increased the need for EA practices that meet high standards and incorporate concrete operational design considerations [4]. In the financial services sector, the need to balance innovation and operational stability presents governance challenges that cannot be addressed by separate architecture or service management practices alone [5, 6].

The selection of TOGAF 10 as the architectural foundation for this study is based on its characteristics as a comprehensive and practical methodology for guiding the development of enterprise IT resources. Unlike the Zachman Framework, which primarily functions as a taxonomic tool for classifying artifacts without providing an execution process, TOGAF offers an iterative and flexible Architecture Development Method (ADM). TOGAF has also achieved a significantly higher performance management score compared to other frameworks such as Zachman (ZEF), FEAF, or RM-ODP. This makes TOGAF a suitable choice for linking business strategy with operational efficiency, particularly in stabilizing SLA metrics and reducing MTTR, which are the primary focus of this research [7].

The operational context of multi-finance organizations illustrates how this gap manifests. Before structured alignment efforts were implemented, incident recovery times showed significant fluctuations, with MTTR reaching 775 minutes in April and SLA compliance falling below 90% during the same period. In addition, operational data showed that manual configuration errors related to system, infrastructure, and database configurations caused 78.57% of incidents, indicating insufficient operational planning and automation during the design phase. These conditions highlight the consequences of architectural decisions that do not incorporate operational performance requirements.

Operational methodologies such as automated observability, cloud monitoring, and Site Reliability Engineering (SRE) have been proven to improve reliability and reduce MTTR [8–9]. Additionally, the use of AIOps (Artificial Intelligence for IT Operations) is emerging as a critical method for automating failure detection and reliability management [2, 10]. However, many organizations adopt these practices as isolated technical solutions while leaving architectural structures unchanged. These limitations reduce long-term effectiveness and increase the need for an integrated design framework. This issue is also reflected in the discourse on AI-supported operational management [11].

Comparable integration efforts in other sectors provide further evidence of the importance of linking architecture with operational performance. Research in transportation systems, manufacturing and supply chain management, fleet management, and digital health demonstrates that when architectural artifacts incorporate explicit operational constraints, organizations achieve greater consistency, reliability, and regulatory alignment [12–14]. These cases collectively confirm that bridging architecture and operations is essential for improving service outcomes.

Previous research indicates that advanced ITSM techniques enhance supply chain efficacy, increase service performance, and stabilize operations at the capability level [15–16]. However, many organizations still use ITSM standards such as ISO/IEC 20000-1 alongside EA frameworks such as TOGAF in parallel. This practice leads to overlapping controls,

disorganized processes, and inconsistent governance structures [17]. Earlier studies call for systematic alignment between ITSM and enterprise-wide architectural processes to eliminate redundancy and improve operational coherence [18–19]. This alignment is crucial because strategic EA directly influences business performance [20], and effective IT governance is essential for decision-making stability in the banking sector [6, 21].

Recurring patterns identified in both the literature and operational data such as variations in MTTR, declines in SLA performance, and a high number of manually caused incidents, demonstrate the importance of an integrated approach that incorporates operability requirements into the architecture lifecycle. Organizations often fail in their EA initiatives due to disconnected implementation strategies [21]. This provides a clear basis for developing an integrated framework that links TOGAF artifacts with ISO/IEC 20000-1 processes. The framework enables organizations to incorporate reliability objectives, estimate SLA performance during the design phase, and improve service resilience.

Therefore, this research seeks to develop an integrated architecture-operational framework that maps TOGAF 10 artifacts directly to ISO/IEC 20000-1:2018 service processes. The objective is to enable measurable improvements in service reliability by embedding design-for-operability principles into the architectural design cycle, thereby bridging the gap between strategic planning and operational execution. The framework is validated through a 14-month Design Science Research (DSR) cycle in a large multi-finance institution, with a specific focus on optimizing SLA and MTTR metrics.

2. Materials and Methods

This section outlined the systematic methodology employed to develop and validate the integrated TOGAF 10 and ISO/IEC 20000-1:2018 framework. The research was grounded in the iterative Design Science Research Methodology (DSRM), which was selected for its suitability in engineering a practical solution to address the gap between architectural design and operational execution. The methodology was implemented through a focused 14-month case study at PT XYZ Multi-finance. The DSRM process adhered rigorously to its six stages, from Problem Identification and Motivation through to the final Communication phase, to ensure the structured development and validation of the core artifacts, namely the Two-Way Traceability Model and the Operability Pattern Catalog. Data collection employed a mixed-methods strategy, combining qualitative insights from stakeholder interviews with quantitative operational metrics (MTTR, MTTD, and SLA compliance) to establish a clear performance baseline and to measure the impact of the implemented solution.

2.1. Study design.

The DSRM was used to design and validate the integrated framework. PT XYZ multi-finance, a financial institution undergoing digital transformation, served as the subject of this 14-month case study. Figure 1 illustrates the Design Science Research (DSR) flow model for the integration of the TOGAF ADM and ISO/IEC 20000-1:2018 service management processes.

Figure 1. DRS methodology flow model for TOGAF ADM and ISO/IEC 20000-1:2018 integration.

As illustrated in Figure 1, this study adopted the DSR methodology, which proceeded through six sequential and iterative steps [22]. The process began with Problem Identification and Motivation, during which the specific gap between strategic architectural planning and operational service execution was diagnosed to establish the practical relevance of the study. This diagnostic phase informed Goal Setting, which defined the objective of developing a solution that embedded ISO/IEC 20000-1 reliability constraints into TOGAF 10 artifacts. Next, the Design and Development phase involved the creation of the proposed artifacts, specifically the Two-Way Traceability Model and the Operability Pattern Catalog. The utility of these artifacts was then validated during the Demonstration phase through a pilot implementation within the live operational environment of PT XYZ Multi-finance. Following the pilot, Evaluation was conducted by quantitatively comparing pre- and post-implementation data related to SLA compliance and MTTR reduction. Finally, the cycle concluded with the Communication phase, in which the resulting framework, empirical findings, and design principles were documented and disseminated to the academic and professional communities.

2.2. Materials and tools.

The main materials used in this study consisted of corporate architecture standards and service management frameworks. TOGAF 10 (The Open Group) was used to guide the development of the enterprise architecture, with a particular focus on the phases of the ADM. ISO/IEC 20000-1:2018 served as the reference standard for ITSM requirements. In addition, several tools and data sources supported the demonstration and analysis stages. The Architecture Repository served as the primary storage for enterprise architecture artifacts, including matrices, diagrams, and catalogs. The organization's ITSM ticketing system was used to obtain historical incident records and service requests as sources of operational data. Furthermore, real-time service availability and latency were monitored using Application Performance Monitoring (APM) tools, ensuring accurate quantitative measurements throughout the assessment period.

2.3. Data collection.

Data collection was conducted using a mixed-methods approach that combined qualitative insights with quantitative metrics to provide a comprehensive understanding of the operational context. This integrated methodology enabled the study to capture both stakeholder perspectives and objective patterns derived from historical performance data. Qualitative data were obtained through structured interviews and participatory observation. Interviews were conducted with key stakeholders, including Enterprise Architects, the Head of IT and Digitalization, and IT Operations Managers, who possessed critical knowledge of system architecture and operational processes. These discussions aimed to identify gaps between system design and operational execution, as well as to validate the relevance and practical applicability of the proposed artifacts. Participatory observation complemented the interviews by providing direct insight into daily operational workflows and decision-making practices,

allowing the identification of contextual factors that might not have emerged through interviews alone.

Quantitative data were collected from historical operational records to establish an analytical baseline. The dataset covered a 14-month period and included incident records, Service Level Agreement (SLA) compliance metrics, and transaction load statistics for two critical digital services: Credit Origination and Digital Collections. These indicators provided evidence of system reliability, service performance, and operational efficiency. The integration of qualitative and quantitative methods was achieved through triangulation, whereby stakeholder insights were used to explain the structural root causes underlying observed performance variations. This dual-stream approach ensured that the resulting framework was both theoretically grounded and operationally feasible within the multi-finance context. Such methodological rigor aligned with the need for systematic integration between ITSM and enterprise architecture to enhance organizational performance.

2.4. The integration procedure was executed in three main stages.

The integration procedure was carried out through three major stages designed to establish alignment between architectural design and operational performance. The first stage, Mapping and Integration, focused on correlating TOGAF ADM phases from the Preliminary phase through Phase H with specific ISO/IEC 20000-1:2018 clauses, particularly those related to Service Level Management, Service Continuity, and Incident Management. This systematic alignment produced a comprehensive Two-Way Traceability Model that linked business value streams to measurable operational key performance indicators (KPIs). The second stage, Artifact Development, involved the creation of key architectural deliverables, including a multi-layer Process Architecture Blueprint (Levels 1–3) and a Catalog of Operability Patterns addressing redundancy, observability, and fault isolation to ensure operational readiness. The final stage, Pilot Implementation, applied the framework to priority digital services by embedding operability checkpoints into the solution architecture and deploying automated recovery workflows. The effectiveness of the integration was evaluated using a before-and-after analysis based on MTTR, SLA compliance rates, and MTTD. To ensure the robustness of the findings, SRE-based root cause analysis was conducted to distinguish improvements resulting from enhanced process discipline from those driven by structural architectural refinements.

3. Results and Discussion

This section began with an examination of the Application Architecture and its interdependencies to provide the groundwork for the proposed solution. It then presented the measured results and corresponding discussion for the established integrated framework. The analysis demonstrated that the Two-Way Traceability Model and the Catalog of Operability Patterns, the two primary research artifacts, directly connected the TOGAF ADM stages with the provisions of ISO/IEC 20000-1:2018, thereby enforcing design-for-operability principles throughout the architectural lifecycle. The resulting operational benefits were demonstrated through quantitative analysis. SLA compliance reached a stable level, MTTR became more consistent, and MTTD decreased substantially. A reduction in manual configuration errors, confirmed through Root Cause Analysis (RCA), further validated the integration and indicated

that the framework effectively bridged the design–operation gap and established an ambidextrous governance model.

3.1. Application architecture and interdependency analysis.

The Application Architecture and Interdependency Analysis was a critical step in this research to identify the structural root causes contributing to the elevated MTTR and MTTD metrics discussed in the Introduction. The initial evaluation revealed that the digital environment of PT XYZ Multi-finance operated within a highly complex ecosystem. The availability of its core digital services depended heavily on a tightly integrated network of upstream channels and downstream core systems. Although this interconnectivity enabled business operations to function effectively, it introduced significant architectural vulnerabilities. The CORE system acted as a central coordination component, effectively becoming a single point of critical failure. This rigid cross-dependency meant that a failure or bottleneck in one domain particularly within the Payment Services layer or the CORE system, immediately triggered cascading failures, propagating downtime across both upstream and downstream systems. To comprehensively visualize these conditions, the architectural analysis focused on mapping the application landscape and tracing data lineage. Figure 2 (Application Landscape Architecture) illustrated the target architecture by mapping the main functional domains, including Payment Channels, Customer Acquisition, and Corporate Administration systems, and by revealing the dependencies among them. This analysis was further complemented by Figure 3 (Conceptual Data Diagram), which delineated the lifecycle of critical data entities from Leads to Contracts and Collections. Together, these analyses confirmed the primary structural causes of the prolonged incident detection and recovery times observed during the pre-implementation phase. The interconnectivity of these domains, as revealed in the architectural analysis, demonstrated a tightly coupled ecosystem in which the CORE System functioned as the singular orchestration point

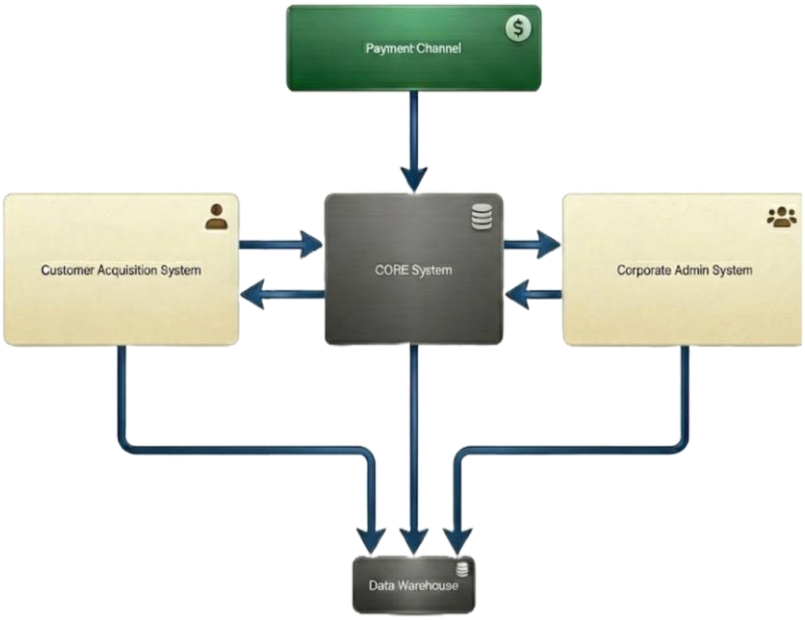

Figure 2. Application landscape architecture.

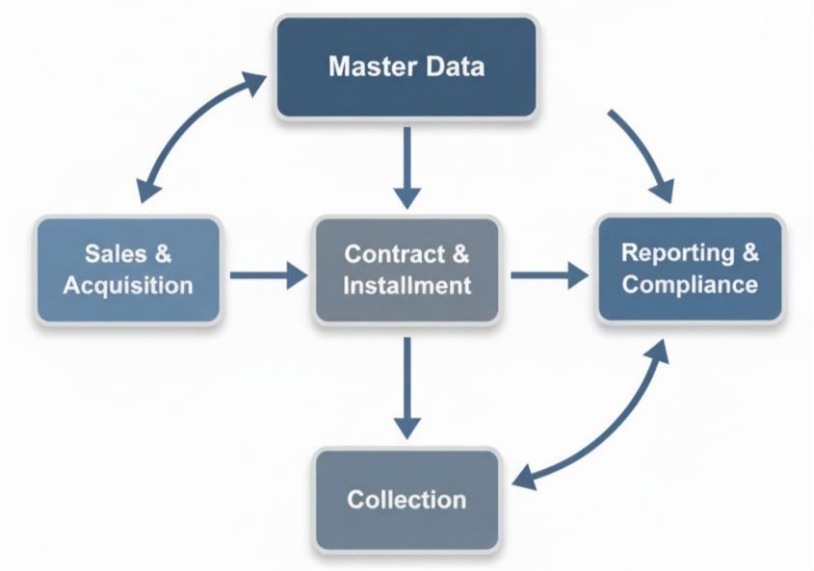

Figure 3. Conceptual Data Diagram.

3.1.1. Payment channel domain.

This domain served as the primary revenue entry point. It aggregated various transaction sources, including the internal Kreditplus Mobile app, retail partnerships (Indomaret, Alfamart), and digital wallets (Dana, LinkAja). Crucially, the detailed topology revealed that all these channels funneled through a single "Payment Services" middleware before reaching the Core. This architecture created a critical dependency: a bottleneck or failure in the "Payment Services" layer immediately severed all incoming revenue streams across all platforms.

3.1.2. Customer acquisition system.

Situated upstream, this domain managed the "Order-to-Cash" value stream. It comprised Sales Activity Apps and Merchant Apps, which fed data into the Loan Origination System (LOS). The bi-directional arrows in Figure 2 represented the constant synchronization required between the acquisition front-end and the Customer Data Management platform. High latency in the Core System directly impacted the ability of sales agents to input new leads or verify customer credit limits in real-time.

3.1.3. Corporate admin system.

This downstream domain handled essential back-office operations. It included the Collection Management System (FieldCol, Deskcall) and Regulator Systems (SLIK). The analysis highlighted that these systems were not autonomous; they relied on real-time data pushes from the Core System to function. For instance, the Collection system could not generate accurate daily call lists if the Core System failed to update payment statuses from the Payment Channel.

3.1.4. Data warehouse integration.

All three domains fed into a centralized Data Warehouse to support Business Intelligence and Reporting. While this provided a unified view for management, the architectural dependency meant that operational incidents in the Core System led to data gaps in executive reporting.

The interdependency findings from the baseline analysis confirmed that while individual modules (such as the LOS or specific Payment Apps) were functionally robust, cross-system dependency management was fragile. The architecture lacked sufficient decoupling mechanisms. Consequently, a failure in the central CORE System or the Payment Services middleware triggered a cascading failure effect, propagating downtime to both the upstream Sales Channels and the downstream Collection Systems. This tight coupling was the primary structural cause of the high MTTD and MTTR observed in the pre-implementation phase.

Complementing this application topology, the analysis examined the underlying data lineage to understand how information propagated through these interconnected domains. This data lifecycle, visualized in Figure 3, illustrated the flow of critical entities from initial "Leads" through to "Contract" execution and "Collection" activities. The flow began in the Sales & Acquisition domain, where potential customer leads were recorded. These leads underwent vetting and were converted into Application Documents. Once approved, this transactional data became foundational Master Data, which included information on customers, dealers, and branches and acted as the unchangeable basis for all subsequent financing operations. The core operational transition occurred when an approved Application was converted into the Contract & Installment entity. This entity functioned as the financial engine of the system, defining the legal structure of the transaction and generating specific Payment Schedules and Settlement records. As indicated in the diagrams, the integrity of the Contract entity depended strictly on accurate references from the upstream Master Data.

Finally, the lineage extended to the downstream domains of Collection and Reporting & Compliance. The Collection workflow was triggered by status changes within the Contract entity, specifically missed payments which initiated recovery actions such as Repossession or Restructure. To comply with regulatory requirements, including credit history filings (SLIK), the Reporting domain simultaneously compiled data from all preceding stages (Sales, Contract, and Collection). This cascading dependency demonstrated that proper Collection and Compliance reporting was immediately impacted in the event of an interruption in upstream Master Data or Contract creation.

3.2. The integrated traceability models.

To mitigate the design–operation gap, this research formulated a Two-Way Traceability Model. The model ensured that architectural decisions in the TOGAF ADM phases were directly linked to operational controls in ISO/IEC 20000-1:2018. Table 1 presents the integration matrix, which served as the core artifact of this research. It mandated that operational metrics, such as SLA targets, were estimated during the design phase rather than being discovered post-deployment. In addition to the traceability matrix, this study implemented a comprehensive Catalog of Operability Patterns integrated into the solution architecture. These patterns were selected to address specific technical deficiencies identified during the baseline analysis and to systematically enhance operational performance. The implementation focused on three key domains: system stability, observability, and automated recovery. To ensure system availability and compliance with the Service Level Agreement (SLA), two critical operability patterns were implemented. First, the Circuit Breaker pattern was deployed at the API Gateway level using a resilience library. This mechanism enabled the system to “fail fast” and return a fallback response during disruptions involving third-party gateways, thereby preventing cascading failures that could result in a complete system outage. Second, the Bulkhead pattern

was applied to mitigate the risk of resource exhaustion. By isolating thread pools for critical functions specifically separating the “Payment” feature from less critical tasks such as “Reporting”, the architecture ensured that priority transactions could continue to be processed even during periods of high resource contention.

Table 1. Integrated traceability matrix (TOGAF ADM x ISO 20000-1).

TOGAF ADM Phase	ISO/IEC 20000-1 Clause	Integrated Activity	Output
Preliminary Phase	Cl. 4.1 Understanding the organization	Defining "Operability" as a core Architecture Principle. Establishing ITSM tools as part of the architecture capability.	Architecture Principles (Service-First)
Phase A: Architecture Vision	Cl. 6.1 Service Portfolio	Aligning business goals with service limits and defining the Service Catalogue.	Service Value Definition & KPI Targets
Phase B: Business Architecture	Cl. 8.2 Service Level Management	Mapping critical business paths (e.g., Lending Process) to availability targets (SLA).	Business Process KPI Map
Phase C: IS Architecture	Cl. 8.7 Service Continuity	Designing data redundancy and API failover mechanisms to support business continuity.	Data Redundancy Schema
Phase D: Tech Architecture	Cl. 8.1 Operational Planning	Selecting technology stacks that support observability (Logs, Metrics, Tracing).	Observability/APM Blueprint
Phase E: Opportunities & Solutions	Cl. 6.3 Service Management Planning	Identifying gaps between current "Manual Ops" and target "Automated Ops". Planning the roadmap for SRE adoption.	Service Improvement Roadmap
Phase F: Migration Planning	Cl. 8.5.2 Service Design & Transition	Scheduling the deployment of the "Observability Stack" ensuring no disruption to live services.	Transition Plan with Rollback Scenarios
Phase G: Implementation Gov	Cl. 8.5 Release & Deployment	Validating the solution against the "Design-for-Operability" checklist before Go-Live.	Deployment Compliance Report
Phase H: Architecture Change Mgmt	Cl. 10.1 Improvement / Cl. 8.6 Change	Using Root Cause Analysis (RCA) data from operations to trigger new architecture cycles.	Architecture Change Request (Based on Incident Data)

To accelerate anomaly identification and reduce MTTD, the architecture addressed the issue of fragmented visibility. A centralized log aggregation pattern was introduced to overcome the challenge of logs scattered across isolated servers. Using log senders such as the New Relic Agent, application logs were sent to a centralized dashboard, significantly reducing the time needed to debug issues. This capability was reinforced by Distributed Tracing, which inserted a unique Trace ID into HTTP headers. This identifier flowed from mobile applications to core systems, allowing operations teams to track specific transactions across microservices and immediately identify the exact location of failures. Automatic recovery techniques were implemented at the database and application layers to systematically reduce MTTR. Container

Auto-Healing was achieved by setting up Kubernetes Liveness and Readiness probes, which enabled "zero-touch recovery" for frozen services by automatically identifying and restarting unresponsive pods. Additionally, Database Self-Correction was implemented through automated scripts designed to detect long-running locks. These scripts automatically terminated blocking sessions to resolve deadlocks, preventing prolonged outages that had previously required manual database restarts.

3.2.1. Technology architecture for observability.

As shown in Figure 4, the Technology Architecture (TOGAF ADM Phase D) was updated to include a dedicated Observability Layer. This layer serves as the technical enabler for ISO/IEC 20000-1 Clause 8.1 (Operational Planning and Control).

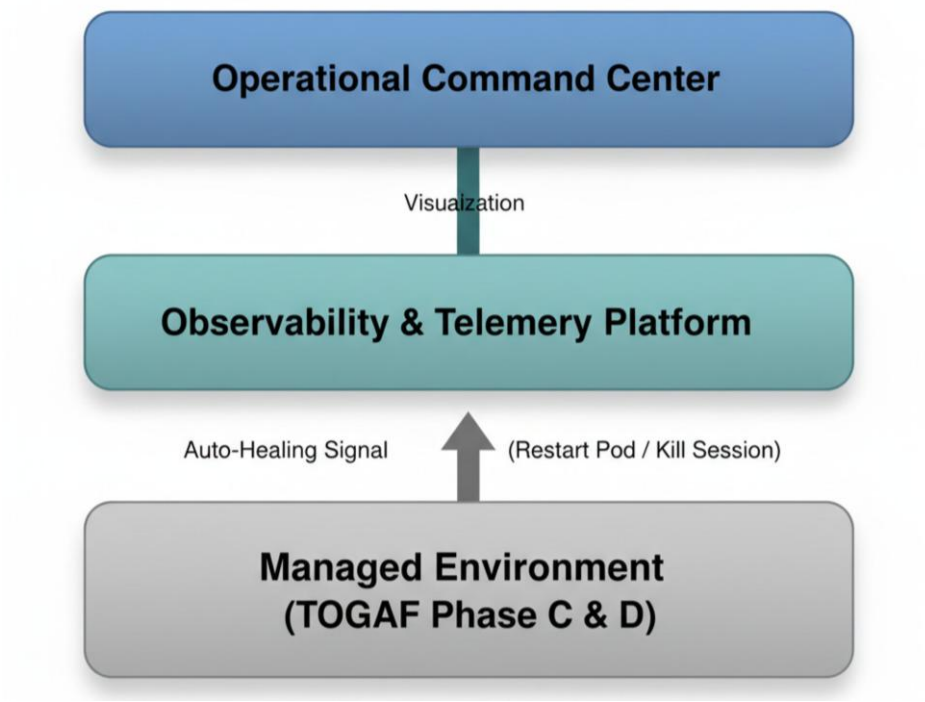

Figure 4. illustrates the technology stack implemented to bridge the gap between application performance and operational visibility.

Unlike the baseline architecture, which relied on decentralized text log files, the target architecture introduced a unified Observability Platform that integrated Centralized Logging, Distributed Tracing (APM), and Automated Alerting. This unified stack provided a cohesive view of system behavior across all application layers, enabling consistent monitoring and faster operational insight. By implementing a unique Trace ID for each transaction, the Operations team could track requests from the API Gateway through intermediate services and ultimately to the Core Database. This end-to-end traceability allowed near real-time identification of root causes whenever performance degradation or system anomalies occurred. Furthermore, the unified observability model reduced reliance on manual log correlation, shortened diagnostic cycles, and enhanced overall service reliability by enabling proactive detection and response to emerging issues.

3.3. Quantitative analysis.

The integration was piloted on two priority services Credit Origination and Digital Collections. The effectiveness of the framework was measured using three key metrics: MTTR, MTTD, and SLA Compliance Rate.

3.3.1. MTTR reduction.

The most significant quantitative improvement observed in this study concerns the MTTR. Defined as the average time required to restore a service to its fully operational state following a failure, MTTR serves as a critical proxy for the system's maintainability.

For the purpose of this analysis, MTTR is calculated using the following equation:

$$MTTR = \frac{1}{n}\sum_{i=1}^{n} (t_{resolve,i} - t_{start,i})$$

Where n is The total number of incidents recorded in the dataset, i is The index denoting a specific incident instance, where $i = 1, \ldots, n.$, $t_{resolve,i}$ is The timestamp when the i-th incident was resolved and service was restored, $t_{start,i}$ is The timestamp representing the actual start of the i-th incident.

To understand the factors driving the recovery performance, Figure 5 decomposes the monthly MTTR into specific issue categories. This analysis is crucial for identifying the root causes of MTTR fluctuations and validating the impact of the integrated framework's implementation. Overall, the data is divided into two clear operational eras: the pre-implementation period, which was marked by high instability. On the other hand, the post-implementation phase demonstrates a fundamental shift toward service stability. This decomposition allows the team to see the specific contribution of each incident type, such as manual configuration errors or third-party issues. Thus, a deep interpretation of the chart will confirm the effectiveness of the "Design for Operability" artifacts in mitigating risk.

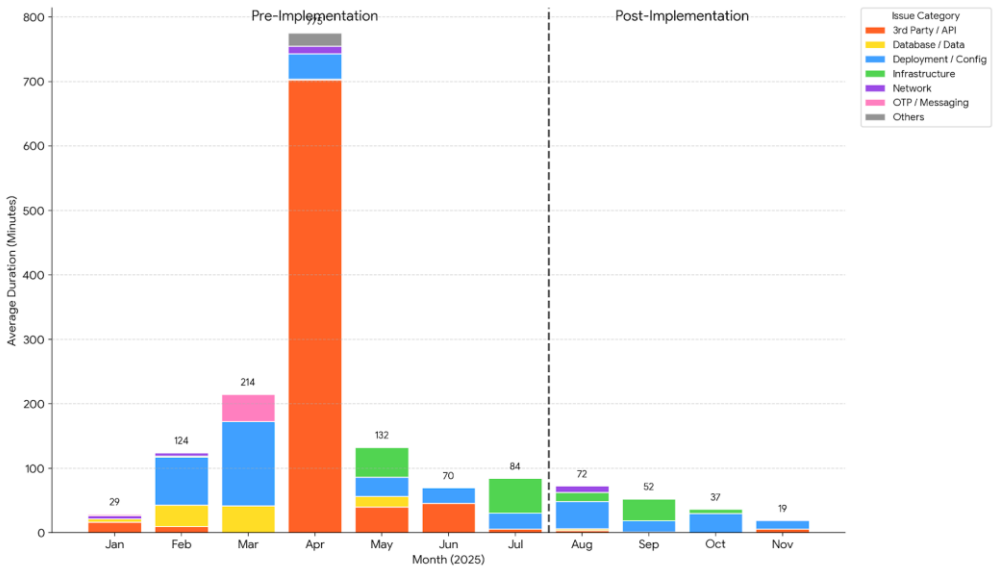

Figure 5. MTTR improvement before and after implementation.

During the pre-implementation phase, MTTR exhibited high volatility driven by two primary factors: external dependencies and configuration issues. As illustrated in Figure 5, the "3rd Party / API Integration" category, represented by the red section, accounted for nearly the entire spike in April, reaching a peak of 775 minutes. This visual evidence confirmed that, prior to architectural integration, the internal team lacked the capability to isolate the system from external vendor failures, resulting in prolonged and excessive downtime. Additionally, the months of March and May showed significant contributions from "Deployment / Configuration" issues (indicated by the blue segment), suggesting that manual release processes and configuration errors were major bottlenecks, often necessitating extensive time for diagnosis and rollback.

In contrast, the post-implementation phase, marked by the vertical dashed line, demonstrated a fundamental shift. Starting in August, the total height of the bars dropped significantly and stabilized below the 60-minute mark. Notably, although technical incidents related to "Infrastructure" and "3rd Party" still occurred, their impact on duration was drastically reduced. This reduction validated the effectiveness of the "Design for Operability" artifacts. The implementation of Circuit Breaker patterns successfully decoupled internal systems from external failures, while the adoption of Automated Release Management (aligned with ISO/IEC 20000-1 controls) eliminated the long recovery times previously associated with deployment errors.

3.3.2. MTTD improvement.

Complementing the reduction in MTTR, the study observed a substantial improvement in the MTTD. MTTD is a critical metric that measures the operational responsiveness of the organization, specifically quantifying the latency between the onset of a service interruption and the moment the IT operations team becomes aware of it. Mathematically, MTTD is calculated using the following formula:

$$MTTD = \frac{1}{n}\sum_{i=1}^{n} (t_{detect,i} - t_{start,i})$$

Where n represents the total number of incidents recorded in the dataset, i denotes the index of a specific incident instance, where $i = 1, \ldots, n$, $t_{detect,i}$ is the timestamp when the i-th incident was detected by the system, and $t_{start,i}$ is the timestamp representing the actual start of the i-th incident.

Prior to the implementation of the integrated framework, the organization relied heavily on manual user reports and decentralized logging systems. This "blind spot" in the architecture resulted in a disproportionately high detection time. As illustrated in Figure 6, during the Pre-Implementation phase (January–July), the average MTTD was 145 minutes, consuming nearly 70% of the total incident lifecycle (204 minutes). In many cases, the team spent more time realizing there was a problem than actually fixing it. However, following the deployment of the Technology Architecture for Observability in August 2025, the detection capability transformed. The introduction of Automated APM Alerting and Distributed Tracing reduced the average MTTD to just 15 minutes.

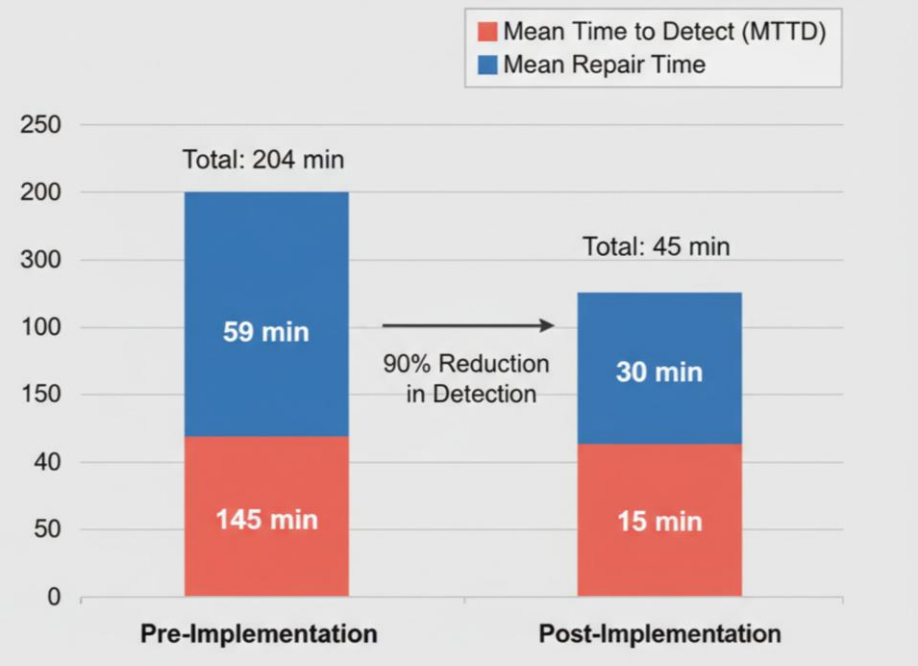

Figure 6. Impact of observability on incident lifecycle: comparison of MTTD and repair time (pre vs. post implementation).

As shown in the chart, while the actual "Repair Time" also improved (from 59 minutes to 30 minutes) due to automated scripts, the most dramatic gain was in detection speed a 90% reduction. This shift confirms that the "Design for Observability" pattern effectively removed the monitoring latency, allowing the operations team to initiate recovery protocols almost immediately after an incident occurs.

3.3.3. SLA compliance rate.

The ultimate measure of the integrated framework's success is its impact on the Service Level Agreement (SLA) compliance rate. Defined as the percentage of time the digital services are available and performing within the agreed operational thresholds, the organization set a stringent target of 99.0% for the fiscal year 2025. To quantify this metric objectively, the study utilized the standard availability calculation formula as defined in the ISO/IEC 20000-1 specification:

$$R_{SLA} = \left(\frac{\sum_{i=1}^{n} 1c_i}{n}\right) x\ 100\%$$

Where n represents the total number of incidents recorded in the dataset, i denotes the index of a specific incident instance, where $i = 1, \ldots, n$, and $1c_i$ is a binary indicator function representing the compliance status of a single incident, with a value of 1 indicating that the SLA was met (compliant) and a value of 0 indicating that the SLA was breached (non-compliant). R_{SLA} represents the aggregate Service Level Agreement compliance rate, expressed as a percentage.

As illustrated in Figure 7, the SLA performance throughout the year tells a story of two distinct operational eras: the volatile pre-implementation phase and the stabilized post-implementation phase. During the first half of the year (January to July), the organization struggled to consistently meet its service targets. The chart reveals a "sawtooth" pattern of

instability, where the SLA hovered between 94% and 96%, consistently missing the 99% threshold. The situation reached its nadir in April, where the compliance rate plummeted to 89.0%. This sharp decline correlates directly with the massive 3rd-party outage discussed in the MTTR analysis, proving that the legacy architecture lacked the resilience mechanisms (such as Circuit Breakers) necessary to isolate the core system from external shocks.

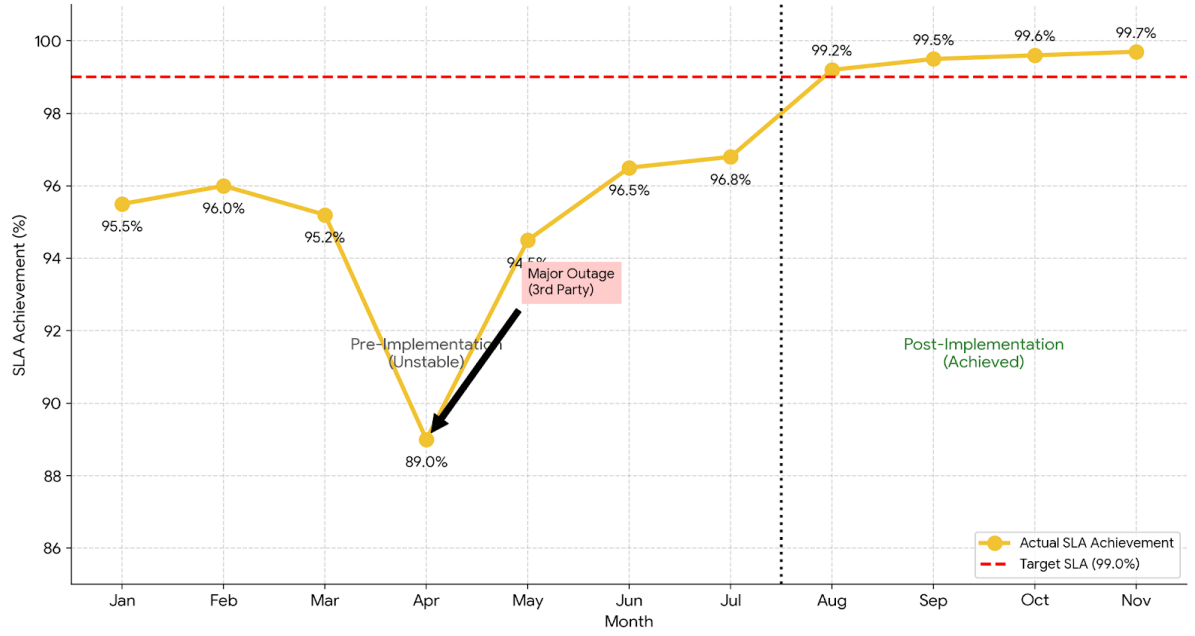

Figure 7. Service Level Agreement (SLA) Compliance Trend (2025).

The turning point occurred in August, coinciding with the full deployment of the ISO 20000-1 aligned processes and TOGAF-based operability patterns. The impact was immediate and profound; for the first time in the year, the SLA breached the target threshold, jumping to 99.2%. This was not merely a recovery but a structural shift in performance. The introduction of "Design for Operability" ensured that minor incidents which previously snowballed into major outages were now auto-remediated by the self-healing container orchestration before they could breach the SLA limits.

Following this initial success, the system demonstrated remarkable durability through the final quarter. From September to November, the SLA compliance rate continued to climb, reaching a peak of 99.7%. This period of sustained high availability confirms that the improvements were not temporary "fixes" but the result of a fundamentally more robust architecture. The stability achieved in these months validates the effectiveness of the Two-Way Traceability Model, which successfully ensured that every architectural change was vetted against strict operational availability criteria before release.

As shown in Figure 7, the system struggled to meet the business target during the first half of the year, with the lowest point in April (89.0%). However, immediately following the deployment of the integrated framework in August, the SLA metric crossed the target threshold, reaching 99.2%. This positive trend continued through November (99.7%), proving that the "Design for Operability" artifacts effectively protected business transactions from technical disruptions.

3.4. Root cause analysis (RCA) and incident composition.

To validate that the improvements in service reliability were driven by architectural interventions rather than external factors, a rigorous SRE-based Root Cause Analysis (RCA) was conducted. This method went beyond simple incident logging by leveraging the unified observability telemetry, specifically Trace IDs and centralized logs, to determine whether a failure originated from manual errors or structural vulnerabilities [2, 23]. The SRE-based approach categorized all recorded incidents into three primary domains: Manual Configuration Errors, Infrastructure/System Anomalies, and 3rd Party/External Issues. This categorization was essential to distinguish improvements resulting from enhanced process discipline from those driven by architectural refinements. Such a data-driven reliability management approach aligns with recent research in AIOps and Service Reliability, which emphasizes reducing operational "noise" through automated failure detection and systematic RCA. To provide a measurable benchmark for these improvements, the study calculated the Incident Volume Reduction Rate (IVRR) using the following formula:

$$IVRR = (\frac{V_{baseline} - V_{current}}{V_{baseline}}) \, x \, 100\%$$

Where $VV_{baseline}$ represents the total volume of incidents recorded during the baseline period, $V_{current}$ represents the total volume of incidents recorded during the current observation period, and IVRR denotes the calculated Incident Volume Reduction Rate, expressed as a percentage.

In the Pre-Implementation phase (January–July), the operational landscape was characterized by high instability, recording a total of 56 incidents as shown in Figure 8. A granular analysis reveals that Infrastructure/System Noise was the largest category with 31 incidents, indicating a fragile infrastructure prone to resource exhaustion and connection timeouts. Manual Configuration Errors (Yellow Segment) accounted for 13 incidents, primarily driven by uncoordinated deployment processes and human error during release cycles, directly pointing to the lack of "Implementation Governance" (Phase G). Furthermore, 3rd Party Issues (Red Segment) accounted for 12 incidents; as noted in the MTTR analysis, these external failures were often catastrophic because the legacy architecture lacked isolation mechanisms.

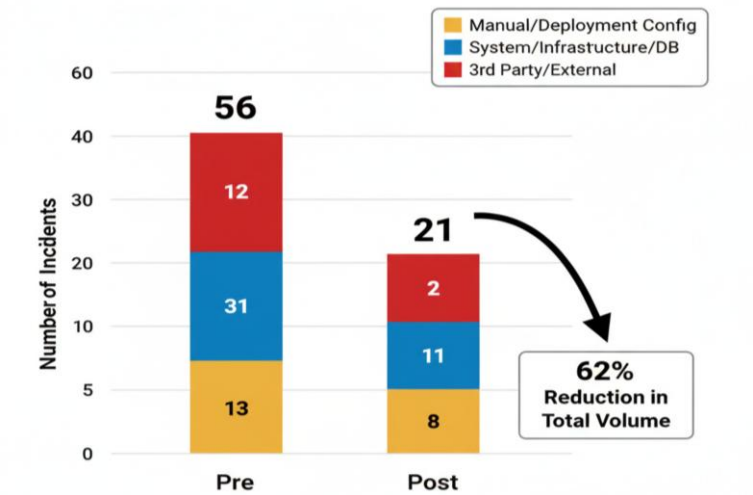

Figure 8. Volume & composition of incidents (pre vs. post implementation), illustrating a 62% reduction in total operational noise.

In the post-implementation phase (August–November), the data demonstrates a fundamental stabilization, with the total volume of incidents dropping to 21 a 62.5% reduction in overall operational noise. This phase saw the elimination of external fragility, where incidents related to 3rd Party factors plummeted from 12 to just 2. This validates the effectiveness of the Circuit Breaker patterns, which successfully shielded the core system from external vendor outages. Additionally, there was a measurable reduction in manual errors, decreasing from 13 to 8. While some deployment issues persist, this reduction confirms that the automated release pipelines and "Design for Operability" checklists are effectively filtering out the majority of human errors before they reach production. In conclusion, the RCA confirms that the improved SLA and MTTR metrics are not accidental but are the direct result of eliminating the bulk of preventable incidents, allowing the operations team to focus on proactive improvements rather than reactive firefighting.

4. Conclusions

The integration of TOGAF 10 artifacts with ISO/IEC 20000-1:2018 processes bridged the gap between high-level enterprise architecture and daily service operations, which had previously caused unstable performance, including an MTTR peak of 775 minutes and low SLA compliance. Conducted using the DSRM at PT XYZ Multi-finance, this study delivered three contributions: a Two-Way Traceability Model linking business value streams to SLA and MTTR outcomes; a Catalog of Operability Patterns addressing redundancy, observability, and fault isolation within the ADM phases; and a prioritization mechanism based on measurable operational impact. Although limited to a set of priority services, the framework demonstrated high generalizability for sectors such as banking, telecommunications, and manufacturing, where IT infrastructure directly affects service reliability. Embedding operational reliability into early architecture phases ensured alignment with SLA targets and enabled resilient digital ecosystems that deliver customer value. The pilot implementation yielded significant operational improvements. MTTR stabilized below 60 minutes, and MTTD dropped from 145 to 15 minutes a 90% reduction attributed to the unified Observability Layer. SLA compliance reached 99.7%. Root Cause Analysis confirmed a 62.5% reduction in incident volume by mitigating external system fragility through Circuit Breaker patterns and minimizing manual configuration errors. In conclusion, the integrated framework effectively aligned strategic architecture with routine service operations, providing measurable and durable performance enhancements. Future practice should institutionalize “operability checkpoints” within architecture governance and adopt a consistent KPI taxonomy (SLA, MTTR, MTTD) to ensure that architectural decisions consistently translate into high-performance digital services.

Acknowledgments

The authors would like to sincerely thank PT XYZ Multi-finance's management and information technology department for their cooperation and permission in carrying out this case study. We would especially want to thank the IT Development team and IT Operations employees who helped with data collecting and offered insightful commentary during the simulations and interviews.

Author Contribution

The authors confirm their contributions to the paper as follows: conceptualization was carried out by Bagus Resa Destian; methodology was developed by Bagus Resa Destian and Panca Dewi Pamungkasari; data collection and data analysis were performed by Bagus Resa Destian; writing of the manuscript was completed by Bagus Resa Destian; and supervision was provided by Panca Dewi Pamungkasari.

Competing Interest

All authors should disclose any financial, personal, or professional relationships that might influence or appear to influence their research.

References

[1] Gong, Y.; Janssen, M. (2023). Why Organizations Fail in Implementing Enterprise Architecture Initiatives? *Information Systems Frontiers*, *25*, 1401–1419. https://doi.org/10.1007/s10796-022-10298-x.

[2] Dang, Y.; Lin, Q. (2024). AIOps: Real-world Challenges and Research Innovations in Service Reliability. *IEEE Software*, *41*(1), 92–99. https://doi.org/10.1109/ICSE-Companion.2019.00023.

[3] Aleatrat, K.; Al-Qirim, N. (2024). Aligning EA and ITSM: A Framework for Continuous Service Improvement. *International Journal of Information Systems and Project Management*, 12(3), 45–60. https://doi.org/10.12821/ijispm120303. .

[4] Sanjurjo, M.; Garcia, F.; Piattini, M. (2024). Enterprise Architecture and IT Governance to Support the BizDevOps Approach: A Systematic Mapping Study. *Information Systems Frontiers*, 27(3), 865–888. https://doi.org/10.1007/s10796-024-10473-2. .

[5] Mulyana, R.; Rusu, L.; Perjons, E. (2024). The key ambidextrous IT governance mechanisms for a successful digital transformation: Case study of Bank Rakyat Indonesia (BRI). *Digital Business*, 4(1), 100083. https://doi.org/10.1016/j.digbus.2024.100083. .

[6] Naguib, M.; Jallow, S.S.; Atif, S.M.; Zaki, M. (2024). The Impact of IT Governance and Data Governance on Financial and Non-Financial Performance. *IEEE Access*, 12, 24567–24582. https://doi.org/10.1186/s43093-024-00300-0. .

[7] Dumitriu, D.; Popescu, M.A.-M. (2020). Enterprise Architecture Framework Design in IT Management. *Procedia Manufacturing*, 46, 932–940. https://doi.org/10.1016/j.promfg.2020.05.011. .

[8] John, L.K. (2025). Optimizing Site Reliability Engineering with Cloud Infrastructure. *International Journal of Computational and Experimental Science and Engineering*, 11(2), 2572–2586. https://doi.org/10.22399/ijcesen.1983. .

[9] Soni, A. (2025). Enhancing Site Reliability Engineering (SRE) Observability: A Comprehensive Approach. *Scholars Journal of Engineering and Technology*, 13(1), 66–68. https://doi.org/10.36347/sjet.2025.v13i01.008.

[10] Pham, L.; Ha, H.; Zhang, H. (2024). Root Cause Analysis for Microservices based on Causal Inference: How Far Are We? *2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Sacramento, CA, USA, 706–718. https://doi.org/10.1145/3691620.3695065.

[11] Putri, N.N.; Mulyana, R.; Adi, T.N. (2025). Ambidextrous AI Governance Design Based on COBIT 2019 Traditional and DevOps for TelCo's Digital Transformation. *TIERS Information Technology Journal*, 6(1), 33–45. https://doi.org/10.38043/tiers.v6i1.6610. .

[12] Parvasi, S.P.; Musavi, M.; Torabi, S.A.; Yap, W.Y.; Lam, J.S.L. (2026). Enhancing transportation service management with bi-level optimization: Competitive pricing and hub location. *European Journal of Operational Research*, 328(3), 815–831. https://doi.org/10.1016/j.ejor.2025.07.033.

[13] Hazen, B.T.; Bradley, R.V.; Bell, J.E.; In, J.; Byrd, T.A. (2017). Enterprise architecture: A competence-based approach to achieving agility and firm performance. *International Journal of Production Economics*, 193, 566–577. https://doi.org/10.1016/j.ijpe.2017.08.022.

[14] Wamema, J.; Alunyu, A.; Amiyo, M.; Nabukenya, J. (2023). Enterprise architecture requirements for standardising digital health in Uganda's health system. *Health Policy and Technology*, 12(4), 100805. https://doi.org/10.1016/j.hlpt.2023.100805.

[15] MacLean, D.; Titah, R. (2023). Implementation and impacts of IT Service Management in the IT function. *International Journal of Information Management*, 70, 102628. https://doi.org/10.1016/j.ijinfomgt.2023.102628.

[16] Lenuwat, P.; Boon-itt, S. (2021). Information technology management and service performance management capabilities: an empirical study of the service supply chain management process. *Journal of Advances in Management Research*, 19(1), 55–77. https://doi.org/10.1108/JAMR-01-2021-0039.

[17] Kornyshova, E.; Deneckère, R. (2022). A Proposal of a Situational Approach for Enterprise Architecture Frameworks: Application to TOGAF. *Procedia Computer Science*, 207, 3493–3500. https://doi.org/10.1016/j.procs.2022.09.408.

[18] Yandri, R.; Suharjito; Utama, D.N.; Zahra, A. (2019). Evaluation Model for the Implementation of Information Technology Service Management using Fuzzy ITIL. *Procedia Computer Science*, 157, 290–297. https://doi.org/10.1016/j.procs.2019.08.169.

[19] Mesquida, A.-L.; Mas, A. (2015). Integrating IT service management requirements into the organizational management system. *Computer Standards & Interfaces*, 37, 80–91. https://doi.org/10.1016/j.csi.2014.06.005.

[20] Ahlemann, F.; et al. (2023). *Strategic Enterprise Architecture Management: Challenges, Best Practices, and Future Developments*. Springer. https://doi.org/10.1007/978-3-642-24223-6.

[21] Singh, P.; Lynch, F.; Helfert, M. (2024). Enterprise architecture for the transformation of public services based on citizen's feedback. *Digital Policy, Regulation and Governance*, 26(1), 38–54. https://doi.org/10.1108/DPRG-11-2022-0123.

[22] Peffers, K.; Tuunanen, T.; Rothenberger, A.M.; Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*, 45–78. https://doi.org/10.2753/MIS0742-1222240302.

[23] Pham, L.; Ha, H.; Zhang, H. (2024). Root Cause Analysis for Microservices based on Causal Inference: How Far Are We? *2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Sacramento, CA, USA, 706–718. https://doi.org/10.1145/3691620.3695065.