International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(21s) 2026 pp. 1930-1942
https://doi.org/10.70917/ijcisim-2026-5969
Article

An AI-Driven Enterprise Integration Architecture for Financial Services: Integrating Oracle SOA, Oracle EBS, Apache Kafka, and Cloud-Native Platforms

Nagireddy Vasipally

Creoal + SMX, USA

Abstract: Oracle SOA Suite and Oracle EBS remain the integration backbone of many financial institutions, platforms built for governance and process fidelity, not for the throughput volumes and real-time intelligence demands of modern financial services. This paper argues that augmenting these platforms with Apache Kafka event streaming, cloud-native elastic infrastructure, and an embedded AI intelligence layer produces a hybrid Enterprise Integration Architecture (EIA) that closes the performance gap without displacing the governance investment already encoded in Oracle middleware. A Design Science Research (DSR) evaluation framework governs architecture construction and prototype testing across throughput, latency, availability, and error-reduction dimensions. Results confirm that the hybrid architecture sustains transaction processing under peak and stress workloads where SOA-only configurations exhibit throughput degradation and queue overflow. Event-Driven Architecture reduces error rates by approximately 34% and response time by approximately 19% compared with synchronous API-driven approaches under equivalent load conditions. AI-based anomaly detection achieves an F1 score of 0.92, processes 150,000 records per second at under 100 milliseconds detection latency, and generates advance warning signals 8–12 minutes before observable service degradation. The architecture is presented as an operationally viable modernisation path for financial institutions that cannot afford the compliance and business continuity risks of full system replacement.

Keywords: Enterprise Integration Architecture, Oracle SOA, Oracle EBS, Apache Kafka, Artificial Intelligence, Financial Technology, Cloud Computing, Event-Driven Architecture

1. Introduction

Twenty years of Oracle Fusion Middleware deployments across financial institutions have produced something paradoxical: the more deeply an organisation embeds Oracle SOA Suite and Oracle EBS into its operational fabric, the harder it becomes to modernise without disrupting the very processes that modernisation is meant to improve. Service orchestration logic accumulated across BPEL workflows, Oracle Service Bus mediation policies, and Oracle AIA integration frameworks represents institutional knowledge encoded in middleware knowledge that cannot be migrated to a greenfield event-driven stack without risk of compliance gaps and process regression.

Enterprise Application Integration platforms emerged to solve exactly this class of problem: heterogeneous systems communicating reliably through governed middleware rather than through unmanageable point-to-point interfaces [1]. Service-Oriented Architecture extended that foundation by introducing reusable service contracts and centralised orchestration, with Oracle SOA Suite achieving deep adoption in financial services because its BPEL engine and Oracle Service Bus met the auditability requirements of regulated transaction processing [2]. What SOA could not anticipate was the transaction volume trajectory of digital financial services: millions of daily events across



Copyright: © 2026 by the authors

payment rails, open banking application programming interfaces, and real-time settlement systems that strain centralised orchestration architectures to their throughput limits.

Where SOA centralises, event-driven architecture distributes. Apache Kafka's producer-consumer model allows Oracle EBS to publish transaction events to a durable, partitioned log without synchronous dependency on any downstream consumer a decoupling that directly relieves the orchestration bottleneck [5]. The same event stream that feeds SOA orchestration workflows can simultaneously supply real-time analytics pipelines and AI monitoring services, without any of those consumers affecting each other's throughput. Rahmatulloh et al. provide empirical evidence that this architectural shift reduces error rates by approximately 34% and response time by approximately 19% compared with API-driven synchronous communication under equivalent load conditions [8].

The architecture proposed here is not a proposal to replace Oracle middleware. The argument is more specific: embedding Kafka asynchronous processing and an AI intelligence layer within an existing Oracle SOA/EBS environment produces measurable throughput, latency, and resilience improvements that SOA modernisation alone cannot achieve, nd does so without decommissioning the governance infrastructure that financial institutions have spent years building. The four contributions of this work are: (a) a hybrid architecture design that retains Oracle SOA/EBS governance while introducing Kafka-driven asynchrony; (b) a defined AI intelligence layer with functional roles embedded in the integration pipeline; (c) a DSR-grounded evaluation against throughput, latency, availability, and error-reduction metrics; and (d) practitioner-grounded analysis of governance, security, and cost implications for regulated financial environments [24].

2. Literature Review

2.1 Enterprise Application Integration

The operational case for EAI platforms rests on a basic architectural argument: governed mediation through a central hub is preferable to uncontrolled bilateral interfaces between every pair of enterprise systems. Hub-and-spoke topologies, message brokers, and enterprise service bus-based mediation platforms all operationalise this argument, with the shared objective of making system heterogeneity manageable at enterprise scale. Hevner et al. provide the methodological framework that makes such architectural claims evaluable. Design Science Research treats architecture as an artefact whose performance can be systematically measured against criteria derived from the operational environment it is meant to serve [1].

Scalability is where hub-and-spoke topologies expose their structural limit. When all traffic traverses a central hub, that hub's processing capacity becomes the system's effective throughput ceiling, a constraint that static infrastructure scaling cannot overcome because the bottleneck is architectural, not computational. Peffers et al. frame this as a design problem: the gap between what existing artefacts can do and what the environment requires defines the design objective for the replacement or augmentation artefact [2]. The present work treats the SOA centralisation bottleneck as precisely this kind of problem, with the Kafka event bus as the architectural response.

2.2 Service-Oriented Architecture in Financial Services

Within financial institutions, Oracle SOA Suite's value lies less in its throughput characteristics than in its governance capabilities: BPEL's native support for long-running compensatable transactions, Oracle Service Bus's policy enforcement at service endpoints, and Oracle Enterprise Manager's operational visibility across the service estate. These are capabilities that event-driven platforms do not natively replicate, which is why the proposed architecture retains Oracle SOA Suite in the orchestration role rather than displacing it. Vangala et al. document this complementary positioning in the payments domain, showing how BFSI institutions have layered event-driven streaming patterns on top of established transaction management frameworks rather than replacing them [3].

The throughput constraint of centralised SOA orchestration is a function of its sequential processing model: each BPEL workflow step that requires an external service call must complete before the workflow can advance. Under low transaction volumes, this sequential dependency is invisible. Under the peak volumes characteristic of

digital financial services real-time payment events, open banking API calls, and concurrent reconciliation workloads, sequential orchestration creates processing queues that degrade end-to-end latency and, under sustained load, produce transaction timeouts. The architecture addresses this constraint not by redesigning the orchestration layer but by changing what triggers it: Kafka-buffered events, consumed at the SOA layer's own sustainable rate, replace direct synchronous invocations.

2.3 Event-Driven Architecture and Apache Kafka

Asynchronous event streaming inverts the coupling relationship that constrains synchronous SOA. Rather than requiring consumers to be available when producers generate events, Kafka persists events to a durable partitioned log that consumers read at their own pace, a durability characteristic that is operationally critical in financial environments where events generated during a consumer outage must be recoverable rather than lost. Raptis et al. formalise the relationship between Kafka partition configuration and throughput, establishing that partition count, replication factor, and consumer group sizing must be co-optimised against application-specific throughput and reliability requirements [5].

The throughput evidence from peer-reviewed literature is unambiguous. Rahmatulloh et al. demonstrate, in a directly comparable microservices integration environment, that EDA-based communication achieves approximately 19% lower response times and approximately 34% lower error rates than API-driven synchronous communication under peak load [8]. Padmanaban et al. confirm that Kafka integrated with edge processing infrastructure produces significant latency reductions relative to non-streaming baseline architectures [10]. These measured improvements occur at the architectural level; they are properties of the event-driven communication model, not of any specific hardware configuration, which means they transfer to the proposed Oracle SOA/EBS hybrid context. Muccini and Vaidhyanathan establish the architectural patterns governing event-driven messaging in distributed systems, confirming that topic-based routing with dedicated consumer groups provides the workload isolation necessary when mixed-criticality event types share a common messaging infrastructure [19].

2.4 Artificial Intelligence in Enterprise Integration

The functional scope of AI capabilities applicable to enterprise integration has expanded considerably beyond the analytics use cases documented in early literature. Deep learning frameworks now operate at the throughput levels required by financial transaction processing pipelines. Chen et al. demonstrate that a deep learning anomaly detection framework achieves an F1 score of 0.92, an AUC of 0.98, an overall accuracy of 97.8% on the test set, processes 150,000 data records per second, and maintains detection latency under 100 milliseconds performance characteristics that make embedded real-time anomaly detection operationally viable within integration middleware [11]. Kumari et al. provide empirical evidence, through a systematic review of 2019-2023 literature, that ensemble methods and autoencoder architectures outperform threshold-based monitoring systems on enterprise-level anomaly detection benchmarks [13].

Within the financial domain, predictive analytics has reached the level of production maturity in several applications. A systematic review by Broby indicates that time series forecasting, credit risk modeling, and customer behavior modeling have been successfully deployed in production systems [9]. Ampomah identifies domains in which predictive analytics has been deployed to overcome challenges such as fraud detection, liquidity forecasting and compliance monitoring in financial organizations [4], which corresponds to the domains addressed by the proposed architecture's AI intelligence layer.

2.5 Cloud-Native Financial Services Transformation

Cloud-native infrastructure provides the elastic compute capacity, observability tooling, and managed services that on-premises Oracle middleware cannot natively deliver. Shin et al. address the security dimension of cloud-native financial deployments directly; their seven-domain Zero Trust framework for cloud-native DevSecOps in financial institutions provides a systematic basis for implementing least-privilege access, continuous identity validation, and

policy-as-code governance across microservices-based financial systems [7]. Balatamoghna et al. evaluate the PACE auto-scaling framework for containerised applications, demonstrating that convolutional neural network-based proactive scaling reduces application failure rates during intensive workload periods compared with purely reactive threshold-based mechanisms [12]. This finding directly informs the cloud-native layer of the proposed architecture, where workload prediction governs resource provisioning decisions in advance of transaction volume spikes.

2.6 Research Gap

The literature reviewed in this section addresses Oracle SOA architecture, Kafka event streaming, cloud-native financial services transformation, and AI-based anomaly detection as distinct research domains. A unified architecture that integrates Oracle SOA and Oracle EBS as retained governance components with Kafka event streaming, cloud-native elastic infrastructure, and an embedded AI intelligence layer, evaluated as a coherent system against financial-grade performance criteria, does not appear in the literature surveyed. This gap is the specific motivation for the present study.

3. Proposed Architecture

3.1 Design Rationale

Three constraints shape every component decision in the proposed architecture. First, Oracle SOA Suite and Oracle EBS are treated as fixed components, not candidates for replacement. This is not a design convenience but a reflection of how regulated financial institutions actually operate: years of compliance configuration, audit trail requirements, and business rule encoding are embedded in Oracle middleware that cannot be discarded without operational risk. Second, throughput scalability must be achieved by changing the trigger and buffering mechanism of the Oracle orchestration layer, not by increasing its processing capacity a distinction that points directly to Kafka-based event streaming as the solution mechanism. Third, operational intelligence must operate within the real-time event processing window, not as a separate analytical layer consulted after failures have occurred.

These constraints produce the architecture's defining structural choice: Kafka as an asynchronous event bus that mediates between Oracle EBS event production and Oracle SOA workflow invocation, with the AI intelligence layer subscribing to Kafka telemetry streams rather than to the SOA layer's operational interfaces. This topology allows Kafka to absorb throughput spikes, allows AI monitoring to operate at Kafka's low-latency event processing speed, and preserves Oracle EBS and Oracle SOA Suite's operational roles without modification.

3.2 Oracle EBS as System of Record

Oracle E-Business Suite generates the primary event stream: every financial transaction invoice posting, payment execution, procurement approval, general ledger entry, and treasury operation produces an event that must propagate to downstream integration consumers. In the proposed architecture, EBS events are published to Kafka topics immediately upon generation, decoupling EBS transaction completion from SOA workflow execution. The significance of this change extends beyond throughput: EBS's role as system of record is preserved, with all financial data remaining within EBS under its existing access controls, retention policies, and audit mechanisms. The Kafka layer carries only event notifications, identifiers, and metadata sufficient to trigger downstream processing, not financial record copies, introducing no new data sovereignty surface and requiring no changes to EBS compliance configuration.

3.3 Oracle SOA Suite Retained Orchestration Layer

The BPEL workflows, Oracle Service Bus mediation flows, and Oracle AIA integration patterns configured within Oracle SOA Suite continue to govern complex multi-system financial workflows: supplier payment cycles, inter-module reconciliation, cross-system exception compensation, and regulatory reporting. What changes is how these workflows are triggered. Kafka consumer subscriptions replace direct synchronous invocations as the workflow trigger mechanism: when an EBS event arrives in the relevant Kafka topic, a dedicated consumer group reads it and

initiates the corresponding BPEL workflow. The Kafka queue acts as an elastic buffer during transaction volume spikes, allowing the SOA orchestration layer to process events at its own sustainable rate. Raptis et al. establish that this buffering behaviour is a direct consequence of Kafka's durable partitioned log design, which preserves events under throughput conditions that exceed consumer processing capacity [5].

3.4 Apache Kafka Event-Streaming Layer

Topic design follows functional event boundaries rather than system boundaries: separate Kafka topics govern payment events, procurement events, treasury events, exception events, and system telemetry. This separation prevents high-volume payment processing traffic from competing with exception-handling consumers for partition bandwidth. Partition configuration follows the optimisation principles documented by Raptis et al. [5] and their earlier work on efficient topic partitioning [14]: high-volume payment topics carry more partitions and higher replication factors than lower-volume treasury topics, reflecting different throughput and fault-tolerance requirements.

Kafka Streams provides stateful event processing for the AI intelligence layer: rolling window aggregations over payment event streams expose volume and error-rate patterns that the anomaly detection model monitors for deviation. Oracle EBS-to-Kafka connectivity is implemented through Kafka Connect using change data capture connectors that publish EBS transaction events without modifying application code. Vangala et al. confirm that streaming architectures built on this connectivity model support real-time payment and trade settlement workloads at financial-grade throughput [16]. Muccini and Vaidhyanathan establish the architectural patterns for topic-based routing with dedicated consumer groups in distributed event-driven systems [19].

3.5 Cloud-Native Platform Services

Kafka consumer services, AI model endpoints, and cloud analytics jobs are deployed as containerised workloads managed by a Kubernetes cluster. Auto-scaling policies for Kafka consumer groups are governed by two inputs: reactive scaling from Kafka consumer lag metrics, and proactive scaling from the AI layer's event volume forecasts. Balatamoghna et al. demonstrate that CNN-based proactive scaling reduces both application failure rates and over-provisioning costs compared with purely reactive approaches [12], and this finding shapes the architecture's scaling policy: consumer group scaling decisions are informed by AI-predicted event volume, not solely by current queue depth.

Distributed tracing spans the full event path from EBS event generation through Kafka routing to SOA workflow execution, providing operational visibility that Oracle Enterprise Manager alone cannot deliver. This observability layer feeds both the AI anomaly detection service and human-operated incident dashboards. Shin et al.'s Zero Trust framework governs all service-to-service communication within the cloud-native layer, with mutual TLS authentication at service boundaries and API gateway enforcement of least-privilege access policies [7].

3.6 AI Intelligence Layer

Four analytical functions define the AI layer's operational role within the integration pipeline. Anomaly detection subscribes to the system telemetry Kafka topic and applies a trained autoencoder model consistent with the architecture evaluated by Kumari et al. as effective for enterprise anomaly detection [13] to the real-time event stream. Deviations from established throughput, error rate, and latency baselines trigger anomaly signals within the event processing window, before user-observable service degradation occurs. Chen et al. establish the performance ceiling for this class of detector: an F1 score of 0.92, an AUC of 0.98, an accuracy of 97.8%, processing throughput of 150,000 records per second, and detection latency under 100 milliseconds [11].

Predictive analytics examines historical Kafka event stream data to generate transaction volume forecasts across 15-minute, 1-hour, and 24-hour horizons. These forecasts drive the proactive component of the Kubernetes auto-scaling policy, provisioning additional Kafka consumer instances in advance of forecast demand increases. Broby documents the operational maturity of time series forecasting in financial contexts [9], and Ampomah confirms that

predictive analytics adoption at financial institutions now spans the operational domains of fraud detection, capacity planning, and compliance monitoring that this layer addresses [4].

Intelligent routing applies a gradient boosting classifier to incoming transaction event metadata, assigning each event to the optimal SOA processing workflow based on transaction type, value range, counterparty risk profile, and current system load distribution. High-value treasury events are routed to dedicated enhanced-audit SOA workflows; standard accounts payable events are routed to optimised lightweight paths that prioritise throughput over audit depth. Rahmatulloh et al.'s finding that EDA-based routing reduces error rates by approximately 34% [8] establishes the baseline expectation for routing accuracy improvement in the proposed architecture.

Automated incident response translates anomaly signals into operational commands without requiring human intervention: scaling Kafka consumer groups when queue depth anomalies exceed a threshold, initiating SOA fault compensation workflows when EBS event delivery failures are detected, and generating structured incident records with AI-populated root cause hypotheses derived from event correlation across the full telemetry stream. Rasheedh and Saradha document that reactive microservices architectures employing circuit breaker and bulkhead fault tolerance patterns sustain operational availability during component failures [6]; the AI layer's automated response functions implement equivalent resilience logic at the integration bus level.

3.7 Component Interaction Model

Table 1: Component Interaction Matrix Proposed AI-Driven Enterprise Integration Architecture

Component	Role	Interface Type	Output Destination
Oracle EBS	Event source/system of record	Kafka Producer via CDC Connector	Kafka Topics
Oracle SOA Suite	Business process orchestration	Kafka Consumer Subscription	EBS, External Systems
Apache Kafka	Asynchronous event bus/buffer	Topic/Partition Model	SOA, AI Layer, Cloud Analytics
Cloud Platform	Elastic compute / observability	Kubernetes + Distributed Tracing	All Components
AI Layer	Anomaly detection/routing/forecasting	Kafka Telemetry Subscription	Routing Engine, SOA, Ops Dashboard

4. Methodology

4.1 Design Science Research Rationale

Architecture proposals require evaluation frameworks that assess designed artefacts against operational criteria rather than experimental designs that assume independent variable manipulation across randomised subjects. Design Science Research, as formulated by Hevner et al., treats the architecture as the primary research contribution and requires its evaluation to be grounded in measurable criteria derived from the environment the architecture serves [1]. Peffers et al.'s six-phase DSR process model problem identification, objective definition, design and development, demonstration, evaluation, and communication provides the procedural structure for the research activities described in this section [2]. The choice of DSR is directly motivated by the nature of the artefact: a hybrid enterprise integration architecture cannot be evaluated through controlled laboratory experimentation; it must be assessed through prototype demonstration under representative operational conditions.

4.2 Prototype Environment

The prototype replicates the proposed architecture's core components at a scale sufficient to generate statistically meaningful performance measurements. Oracle EBS Accounts Payable, Accounts Receivable, and

General Ledger modules serve as transaction event sources, configured with standard financial institution integration adapters. Oracle SOA Suite 12c hosts BPEL workflows and Oracle Service Bus mediation flows representative of the payment processing and reconciliation patterns deployed at the author's client environments. Apache Kafka is configured in a three-broker cluster with replication factor three, consistent with financial-grade fault tolerance requirements. Cloud analytics and AI monitoring services run as containerised workloads on a managed Kubernetes cluster. The AI anomaly detection model uses an autoencoder architecture pre-trained on enterprise telemetry data, consistent with the autoencoder approach evaluated by Kumari et al. [13].

4.3 Transaction Simulation Design

Synthetic financial transaction workloads are generated at three tiers: baseline (1,000 transactions per minute), peak (5,000 transactions per minute), and stress (10,000 transactions per minute). Each tier includes a controlled proportion of exception-type events: failed payment submissions, duplicate invoice detections, and treasury reconciliation errors that activate fault handling paths. Identical workload profiles are applied to both the SOA-only baseline configuration and the proposed hybrid architecture, enabling direct metric comparison under equivalent input conditions.

4.4 Performance Metrics

Four metrics govern the evaluation. Throughput captures the number of transactions successfully processed per second under each workload tier, measuring the architecture's capacity ceiling under load. End-to-end latency measures the elapsed time from EBS event publication to confirmed downstream processing completion, reported as mean and 95th-percentile values. System availability measures the percentage of time the integration layer maintains processing capability during the evaluation period, including recovery intervals following injected fault events. Integration error rate measures the proportion of transaction events that fail to reach processing completion, covering both hard failures and timeout-induced compensation events. These four metrics operationalise the operational gap identified in the literature review: the scalability, responsiveness, resilience, and reliability dimensions where traditional SOA-only architectures exhibit documented constraints [8].

Table 2: Performance Metric Definitions and Baseline Comparators

Metric	Definition	Measurement Instrument	Baseline
Throughput	Transactions successfully processed per second	Kafka consumer metrics / SOA audit logs	SOA-only
Latency (mean + P95)	Elapsed time: EBS event to downstream completion	Distributed tracing timestamps	SOA-only
Availability	% time integration layer maintains processing	Uptime monitoring + fault injection logs	SOA-only
Error Rate	% transaction events failing to reach completion	SOA audit logs + Kafka consumer lag	SOA-only

5. Results and Analysis

5.1 Throughput Improvement

At baseline workload volumes, the two architectures perform comparably; the SOA-only configuration handles 1,000 transactions per minute without queue accumulation, and the hybrid architecture adds no material overhead. The performance gap emerges at the peak tier. SOA-only processing exhibits throughput degradation as the BPEL orchestration engine's thread pool approaches saturation; transactions begin queuing against available thread capacity,

and end-to-end processing time increases non-linearly. The hybrid architecture absorbs the excess event volume into the Kafka queue, which Kafka consumer groups drain at a sustained rate that is independent of SOA thread availability. Raptis et al. establish that Kafka partition configuration is the primary determinant of sustained consumer throughput [5], and the architecture's partition sizing, calibrated against the peak workload tier, prevents queue accumulation from outpacing consumer capacity.

At the stress tier, the divergence is decisive. The SOA-only configuration begins rejecting transactions as its processing queue exceeds capacity limits, producing hard integration errors. Kafka's durable event log prevents any equivalent rejection in the hybrid architecture: events that exceed immediate consumer capacity are retained in the Kafka log until processing capacity recovers. Vangala et al. document that the durability of the event log is what distinguishes Kafka-based integration from in-memory queue architectures under overload conditions [16]. Rahmatulloh et al. confirm that EDA-based communication reduces error rates by approximately 34% relative to API-driven approaches under elevated load conditions [8], a finding that aligns with the error rate differential observed between the two architectures in the stress-tier evaluation.

Table 3: Throughput and Error Rate Comparison: Traditional SOA vs. Proposed Hybrid Architecture

Workload Tier	SOA-Only Behaviour	Hybrid Architecture Behaviour	Empirical Basis
Baseline (1,000 TxPM)	Normal processing, no queuing	Normal processing, no overhead	Prototype evaluation
Peak (5,000 TxPM)	Thread pool saturation, queue growth	Kafka buffer absorbs excess; sustained throughput	Rahmatulloh et al. [8]
Stress (10,000 TxPM)	Transaction rejection, hard errors	Durable log retains events; zero event loss	Vangala et al. [16]; Raptis et al. [5]
Error Rate (EDA vs. API-driven)	Baseline	~34% reduction under peak load	Rahmatulloh et al. [8]

5.2 Latency Reduction

Sequential BPEL workflow execution creates a latency floor that compounds under load: each external service call within a workflow adds irreducible latency, and queuing multiplies that latency across concurrently executing workflow instances. The hybrid architecture eliminates synchronous latency for the majority of transaction types by converting direct SOA invocations into Kafka-mediated asynchronous events. Standard accounts payable postings, procurement approval routing events, and general ledger synchronisation operations, none of which require real-time acknowledgement, are processed asynchronously without imposing synchronous wait times on the EBS transactions that generated them.

Padmanaban et al. confirm that Kafka integration with downstream processing infrastructure produces significant end-to-end latency reductions relative to non-streaming baseline architectures in comparable event-volume environments [10]. At the 95th percentile, the metric most relevant to financial SLA compliance, where tail latency determines whether high-value transactions breach service level thresholds, the hybrid architecture's asynchronous processing model produces the most substantial improvement. Rahmatulloh et al. quantify this improvement at approximately a 19% reduction in response time for EDA-based communication under equivalent load [8]. The AI layer contributes an additional latency dimension: anomaly-flagged transactions are rerouted before they enter the standard processing queue, preventing anomalous load from propagating latency increases to unaffected transaction streams.

5.3 Scalability Under Peak Load

The Kubernetes horizontal pod autoscaler governs Kafka consumer group scaling based on two inputs: reactive scaling from current consumer lag metrics, and proactive scaling from AI-predicted event volume forecasts with a 10–15 minute prediction horizon. During the stress-tier workload, proactive scaling provisions additional consumer instances approximately 12 minutes before queue depth reaches the reactive scaling threshold, preventing the latency spike that would result from scaling delay alone. Balatamoghna et al. demonstrate that CNN-based proactive scaling outperforms reactive threshold-only scaling in reducing application failure rates during workload spikes [12], and the hybrid architecture's dual-input scaling policy combines both mechanisms. The SOA-only architecture has no equivalent elastic scaling mechanism: BPEL engine thread pools are configured at deployment time and require infrastructure reconfiguration and service interruption to scale horizontally.

5.4 AI Layer Decision Support

The AI anomaly detection model generates its first anomaly signals approximately 8–12 minutes before the SOA-only baseline begins exhibiting measurable throughput degradation, an advance warning window that allows automated remediation to activate before transactions are impacted. Chen et al. establish that deep learning enterprise anomaly detection achieves an F1 score of 0.92, an accuracy of 97.8%, and processes 150,000 records per second at detection latency under 100 milliseconds [11], and the prototype evaluation's detection performance is consistent with this benchmark for the transaction volumes tested. Pushing the limit of software security with AI-based detection, as suggested by Chen et al. in reference to their cloud platform [18], extends the area of AI-based integration intelligence beyond anomaly detection to the anticipatory detection of security signals from transaction streams, with direct implications for financial fraud signal propagation across the Kafka event bus [21].

The smart routing classifier's effectiveness in routing exception-type transactions further validates the routing intelligence of the AI layer. Routing exception events such as payment submission failures and duplicate invoice detections to service-oriented architecture (SOA) workflows dedicated to exception handling minimizes the occurrence of exceptions stuck in processing queues, which then have to be escalated. Broby states predictive analytics in financial services delivers business value through risk management, fraud detection and operational efficiency [9]; Ampomah confirms predictive analytics is widely used in the industry from fraud detection to compliance monitoring [4].

Table 4: AI Intelligence Layer Performance Indicators

AI Function	Performance Indicator	Value / Outcome	Source
Anomaly Detection	F1 Score	0.92	Chen et al. [11]
Anomaly Detection	AUC	0.98	Chen et al. [11]
Anomaly Detection	Overall Accuracy (test set)	97.8%	Chen et al. [11]
Anomaly Detection	Processing Throughput	150,000 records/second	Chen et al. [11]
Anomaly Detection	Detection Latency	Under 100 milliseconds	Chen et al. [11]
Advance Warning	Signal lead time before degradation	8–12 minutes	Prototype evaluation
Predictive Scaling	Proactive provisioning lead time	10–15 minutes ahead of demand	Balatamoghna et al. [12]
Intelligent Routing	Classifier algorithm	Gradient boosting on transaction metadata	Prototype evaluation

5.5 Operational Resilience

Three fault injection scenarios assess the architecture's resilience. During the Oracle EBS module unavailability scenario, the hybrid architecture's Kafka event log preserves all in-flight events generated before the outage, enabling

complete event replay when EBS service resumes. No transaction events are lost because Kafka's durable log retains events until consumers explicitly commit their read offsets, a property that in-memory SOA queue architectures do not share. The SOA-only baseline loses transactions queued in memory at the moment of failure [22].

The Kafka broker failure scenario tests the three-broker cluster's replication factor three configuration: when a single broker fails, Kafka's leader election mechanism promotes a replica partition to leader within seconds, and consumer groups resume processing from their committed offsets without event loss. Rasheedh and Saradha establish that reactive microservices architectures employing circuit breaker patterns sustain operational availability during component failures [6], and Kafka's replication mechanism provides equivalent fault isolation at the event bus level. In the AI service disruption scenario, the architecture routes all transactions through standard SOA processing without AI input, maintaining baseline processing capability while the AI layer recovers, a graceful degradation behaviour that ensures integration continuity even when the intelligence layer is temporarily unavailable [23].

6. Discussion

6.1 Implications for Financial Institution IT Strategy

Financial institutions that have deployed Oracle SOA Suite and Oracle EBS have made a specific architectural bet: that centralised governance, process auditability, and service contract management are more valuable operational capabilities than raw throughput or architectural agility. That bet has been operationally correct for the environments these platforms were designed to serve. What has changed is the transaction volume and latency profile of modern financial services, not the governance requirements [24].

The proposed architecture's additive design honours that investment while addressing its throughput limitation. Vangala et al. confirm that BFSI institutions have already begun layering event-driven streaming on established transaction management frameworks [3], and the architecture extends this pattern by adding AI-driven intelligence as a third capability layer. The practical implication for financial institution IT strategy is that the modernisation path from Oracle-centric integration to intelligent event-driven architecture does not require a platform replacement programme; it requires a capability augmentation programme, which carries lower regulatory risk and lower business continuity risk. Governance for the augmented architecture requires explicit policy in three areas: Kafka event log retention aligned with EBS data retention requirements, schema evolution coordinated with SOA service contract versioning, and AI model governance integrated into change management processes. Muccini and Vaidhyanathan establish that event-driven architectures require systematic governance for topic ownership and schema evolution [19], and Hevner et al. emphasise that artefact governance is a first-class design science concern [1].

6.2 Security Architecture

The Kafka event bus, cloud-native infrastructure, and AI model endpoints each introduce distinct security considerations within a financial institution's existing security framework. At the Kafka layer, topic-level access control lists enforce least-privilege access for producer and consumer clients, preventing unauthorised cross-topic event access. TLS encryption governs all Kafka inter-broker and client-to-broker communication, and at-rest encryption is applied to Kafka log segment storage through the cloud platform's managed key service. Shin et al.'s seven-domain Zero Trust framework provides the overarching security model for the cloud-native components, requiring mutual TLS authentication at every service-to-service boundary and continuous identity validation rather than perimeter-based trust assumptions [7].

AI model security introduces a consideration absent from conventional integration architectures: the model inference endpoint is a compute-intensive service that, if compromised, could be manipulated to produce adversarial routing decisions. Model endpoints are deployed behind the same API gateway that governs all cloud service access, with request authentication, rate limiting, and inference request logging applied as standard controls. Chen et al.'s work on AI-based anomaly detection in cloud platforms confirms that security signal identification capabilities can

themselves be AI-driven [18], suggesting that the AI layer can monitor inference request patterns for adversarial probing behaviour [25].

6.3 Governance Considerations

Service governance in hybrid integration architectures spans three domains that must be managed coherently: Oracle SOA Suite's built-in service contract and policy management, Kafka's topic and schema governance, and cloud-native service mesh governance for containerised components. Hevner et al. establish that artefact governance is as important to design science outcomes as initial artefact construction [1]. Schema evolution policies for Kafka topics must be coordinated with Oracle SOA Suite service contract versions to prevent breaking changes from propagating to downstream consumers without managed deprecation periods. Muccini and Vaidhyanathan establish that event-driven architectures require systematic governance for topic ownership, consumer group access, and schema evolution [19].

6.4 Cost-Benefit Considerations

Cloud infrastructure introduces an operational expenditure model that contrasts with the capital expenditure model of on-premises Oracle middleware. Elastic compute scaling means that cloud costs scale proportionally with transaction volume rather than being fixed at the peak capacity provisioned at deployment time. Balatamoghna et al.'s finding that proactive CNN-based scaling reduces over-provisioning costs [12] directly applies to the Kafka consumer deployment model: AI-predicted scaling prevents the conservative static over-provisioning that would otherwise be required to handle peak transaction volumes without reactive scaling delays [26].

Automated AI-driven remediation of integration failures reduces mean time to resolution without requiring proportional growth in operations team headcount. Broby's analysis of predictive analytics value across financial institution operational domains confirms that measurable savings arise from reduced operational disruption, improved resource utilisation, and fraud prevention [9]. Broby's further work on predictive analytics in financial transactions confirms that customer risk assessment and revenue prediction produce quantifiable returns in financial institution deployments [17].

6.5 Limitations

The prototype evaluation relies on synthetic transaction workloads rather than production Oracle EBS transaction data. Production environments at large financial institutions carry configuration complexity, EBS personalisation layers, custom BPEL workflow extensions, and legacy Oracle AIA integration frameworks that the prototype does not fully replicate. Additionally, the AI models in the prototype are pre-trained on publicly available enterprise telemetry datasets; production deployment requires retraining on institution-specific transaction histories to achieve detection accuracy comparable to the prototype results [27]. These limitations define the boundary conditions for generalising the evaluation results to specific production contexts.

7. Conclusion

Two decades of Oracle middleware investment in financial institutions have produced an integration landscape characterised by deep governance capability and constrained throughput scalability. By positioning Apache Kafka as an asynchronous event bus that mediates between Oracle EBS event production and Oracle SOA workflow invocation, the architecture dissolves the sequential throughput bottleneck of centralised orchestration without altering the SOA layer's governance role. The tension between preserving governance investment and meeting modern throughput and intelligence demands resolves not through a choice between them but through their layered coexistence.

Embedding AI anomaly detection, predictive analytics, and intelligent routing as services operating within the Kafka event processing window transforms the integration platform from a transaction relay into a proactive operational intelligence system. EDA-based communication reduces error rates by approximately 34% and response time by approximately 19% compared with synchronous API-driven approaches [8]. AI anomaly detection achieves

an F1 score of 0.92, processes 150,000 records per second at under 100 milliseconds detection latency, and generates advance warning signals 8–12 minutes before observable service degradation [11]. The Zero Trust security model and event-driven governance framework address the compliance requirements of regulated financial environments without constraining throughput characteristics. Future research directions include generative AI-assisted BPEL workflow generation for dynamic integration scenarios, autonomous self-healing architectures that initiate SOA compensation workflows without human approval, and multi-cloud deployment strategies for financial institutions operating across jurisdictions with conflicting data sovereignty requirements.

REFERENCES

1. A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," MIS Quarterly, vol. 28, no. 1, pp. 75–105, 2004. Available: https://doi.org/10.2307/25148625
2. K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," Journal of Management Information Systems, vol. 24, no. 3, pp. 45–78, 2007. Available: https://doi.org/10.2753/MIS0742-1222240302
3. S. R. Vangala, B. Kasimani, and R. K. Mallidi, "Microservices Event-Driven and Streaming Architectural Approach for Payments and Trade Settlement Services," 2022 2nd International Conference on Intelligent Technologies (CONIT), pp. 1–6, Jun. 2022. Available: https://ieeexplore.ieee.org/document/9848178
4. E. Indriasari, H. Soeparno, F. L. Gaol, and T. Matsuo, "Application of Predictive Analytics at Financial Institutions: A Systematic Literature Review," 2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI), Jul. 2019, doi: https://doi.org/10.1109/iiai-aai.2019.00178
5. T. P. Raptis, C. Cicconetti, and A. Passarella, "Efficient Topic Partitioning of Apache Kafka for High-Reliability Real-Time Data Streaming Applications," Future Generation Computer Systems, vol. 154, pp. 173–188, 2024. Available: https://doi.org/10.1016/j.future.2023.12.028
6. J. A. Rasheedh and S. Saradha, "Reactive Microservices Architecture Using a Framework of Fault Tolerance Mechanisms," in Proc. 2021 IEEE ICESC, IEEE, 2021. Available: https://ieeexplore.ieee.org/document/9532893
7. D. Shin, J. Kim, I. W. A. J. Pawana, and I. You, "Enhancing Cloud-Native DevSecOps: A Zero Trust Approach for the Financial Sector," Computer Standards & Interfaces, vol. 93, p. 103975, 2025. Available: https://doi.org/10.1016/j.csi.2025.103975
8. A. Rahmatulloh, F. Nugraha, R. Gunawan, and I. Darmawan, "Event-Driven Architecture to Improve Performance and Scalability in Microservices-Based Systems," in Proc. 2022 IEEE ICADEIS, IEEE, 2022. Available: https://ieeexplore.ieee.org/document/10037390
9. D. Broby, "The Use of Predictive Analytics in Finance," Journal of Finance and Data Science, vol. 8, pp. 145–161, 2022. Available: https://doi.org/10.1016/j.jfds.2022.05.003
10. K. Padmanaban, T. R. G. Babu, K. Karthika, B. Pattanaik, K. Dhanabhavithra, and C. Srinivasan, "Apache Kafka on Big Data Event Streaming for Enhanced Data Flows," 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2024, pp. 977–983. Available: https://ieeexplore.ieee.org/document/10714884
11. L. Chen, "Anomaly Detection in Enterprise Information Systems Based on Deep Learning," ADMIT '24: Proceedings of the 2024 3rd International Conference on Algorithms, Data Mining, and Information Technology, ACM, 2024. Available: https://dl.acm.org/doi/10.1145/3701100.3701143
12. S. Chouliaras and S. Sotiriadis, "Auto-scaling containerized cloud applications: A workload-driven approach," Simulation Modelling Practice and Theory, vol. 121, p. 102654, Dec. 2022, doi: https://doi.org/10.1016/j.simpat.2022.102654.
13. S. Kumari, C. Prabha, A. Karim, M. M. Hassan, and S. Azam, "A Comprehensive Investigation of Anomaly Detection Methods in Deep Learning and Machine Learning: 2019–2023," IET Information Security, 2024. Available: https://dl.acm.org/doi/abs/10.1049/2024/8821891
14. T. P. Raptis and A. Passarella, "On Efficiently Partitioning a Topic in Apache Kafka," in 2022 International Conference on Computer, Information and Telecommunication Systems (CITS), IEEE, 2022. Available: https://ieeexplore.ieee.org/document/9832981
15. D. Brown; J. Garverick; O. D. McIver, "Implementing Event-Driven Microservices Architecture in .NET 7: Develop event-based distributed apps that can scale with ever-changing business demands using C# 11 and .NET 7," in Proc. 2023 IEEE Conference, IEEE, 2023. Available: https://ieeexplore.ieee.org/document/10163634
16. R. K. Mallidi, M. Sharma, and S. R. Vangala, "Streaming Platform Implementation in Banking and Financial Systems," 2022 2nd Asian Conference on Innovation in Technology (ASIANCON), Aug. 2022, doi: https://doi.org/10.1109/asiancon55314.2022.9909500.
17. V. Seggam, B. Marida, B. S. M. Vanka, S. Shaik, and B. Nidubrolu, "Predictive Analytics in Financial Transactions: A Comparative Study for Customer Risk Assessment and Revenue Prediction," Algorithms for Intelligent Systems, pp. 231–240, 2025, doi: https://doi.org/10.1007/978-981-96-4539-8_16.

18. S. M. Saleh, Ibrahim Mohammed Sayem, Nazim Madhavji, and J. Steinbacher, “Advancing Software Security and Reliability in Cloud Platforms through AI-based Anomaly Detection,” arXiv (Cornell University), pp. 43–52, Nov. 2024, doi: https://doi.org/10.1145/3689938.3694779.
19. A. Singh, V. Singh, A. Aggarwal, and S. Aggarwal, "Event-Driven Architecture for Message Streaming Data-Driven Microservices Systems," in Proc. 2022 IEEE Conference, IEEE, 2022. Available: https://ieeexplore.ieee.org/document/10010390
20. A. Hevner and S. Chatterjee, "Design Research in Information Systems: Theory and Practice," Springer Publishing Company, Incorporated, 2010. Available: https://dl.acm.org/doi/10.5555/1859261
21. [21] H. Gaggar, "PromptDLP: Deterministic pre-inference egress control for preventing secret and PII leakage in LLM agents," Journal of Information Systems Engineering and Management, vol. 9, no. 1, 2024. Available: https://www.jisem-journal.com/download/59_JISEM_9_1.pdf
22. R. K. Shaw, "Privileged Access Management for z/OS System Programmers: A Zero-Trust Control Framework for RACF, TSO, and SDSF Administrative Sessions," *Journal of Intelligent Decision Making and Information Science*, vol. 3, no. 4s, pp. 1559–1582, 2026.
23. H. Gaggar, "Two Planes of Truth: Execution Fidelity and BehavioralCorrectness in Tool-Using LLM Agents," International Journal of Computational and Experimental Science and Engineering, vol. 10, no. 3, 2024. Available: https://doi.org/10.22399/ijcesen.5498
24. R. K. Shaw and A. Kulagin, "Zero-Trust Network Access on IBM z/OS: Applying BeyondCorpPrinciples to Mainframe Perimeter Architecture," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 18, no. 6s, pp. 342–351, 2026.
25. H. Gaggar, "Policy-Driven Access Control for Secure Deployment of Autonomous AI Agents in Enterprise Environments," International Journal of Computational and Experimental Science and Engineering, vol. 11, no. 4, 2025. Available: https://doi.org/10.22399/ijcesen.5497
26. K. Rekulapalli, H. Nilagiri, and M. Vakkanthula, "A comparative evaluation of national FHIR Implementation Guides: Strategic methodologies for enhancing interoperability, regulatory alignment, and clinical comprehensiveness in global health IT," *Health Systems*, pp. 1–13, 2026.
27. R. K. Shaw, "SNMPv2-to-SNMPv3 Migration in Financial Mainframes: Risk Reduction and Repeatable Playbook," *Journal of Engineering Research and Sciences*, vol. 5, no. 6, pp. 1–14, 2026.