Real-Time Claims Processing Using Event-Driven Architectures

(Author Details)
Anjani Haritha Sannidhanam*
Graduate Researcher, Syracuse University, USA

Abstract

In today's fast-paced insurance and financial landscape, where speed, accuracy, and automation are paramount, real-time claims processing is a feature that can significantly impact customer satisfaction and operational efficiency. However, traditional batch-type processing systems can face challenges such as latency, scalability, and slow decision-making, especially in high-volume settings. Event-Driven Architectures (EDA) offer a game-changing solution, allowing the system to respond immediately to new events like claims submissions, verification updates, and fraud alerts. Asynchronous communication and decoupling of system components enables EDA to facilitate continuous data flow, real-time analytics and automated decision-making across distributed services. This architectural model offers greater responsiveness, better system scalability and greater fault-tolerance in dynamic operational environments. In addition, the stream processing and event correlation functionality helps speed fraud detection and streamlines claims validation. Overall, EDA provides a solid basis for developing new, adaptive, efficient and resilient real-time claims processing systems.

Keywords: Event-Driven Architecture, Real-Time Processing, Claims Management, Distributed Systems, Stream Processing, Fraud Detection, Microservices, Scalability

DOI: 10.21590/07.01.02

I. Introduction

Digital enterprises are constantly evolving with the need for real-time decision-making, which has led to the development of systems capable of handling large volumes of data at high speeds with minimal latency. For insurance claims management, for example, batch processing is becoming too slow to meet the need for prompt validation, fraud identification and settlement processing. To tackle these limitations, Event-Driven Architectures (EDA) have become a fundamental paradigm to handle business events as they happen, instead of in scheduled periods (Emily & Oliver, 2020).

A fundamental basis of EDA is the idea of event production, detection, routing and consumption in loosely coupled services. This design allows for non-synchronous communication among the various pieces of the system, which can enhance scalability, responsiveness, and fault tolerance. The first conceptual development of real-time event-driven systems proved successful in monitoring activities and early warning systems, and paved the way for the development of real-time systems in enterprise systems (Li, 2005).In later developments, these principles were extended to distributed business intelligence systems, as they rely on event-driven mechanisms to support real-time execution of processes across supply chains and enterprise processes (Linden et al., 2010).

At the same time, the coupling of event-driven principles with service-oriented and distributed architectures, has increased their usefulness in complex architectures. Event-driven models have been identified as having a significant role in augmenting the service-oriented architectures, with a view to increasing modularity and facilitating dynamic interactions between systems (Malekzadeh, 2010; Clark & Barn, 2012). Moreover, with the development of real-time analytics, it has been shown that event-driven systems are capable of analyzing continuous data sources to provide immediate insights and enrich business intelligence capabilities of next generation (Erik & Emma, 2018; Braun et al., 2015).

EDA is an essential component in the processing of data streams that are generated continuously from sensors in modern distributed systems, like Internet of Things (IoT) and geospatial infrastructures. In the field of edge analytics and IoT gateways, it facilitates localized processing and minimizes latency, particularly for edge computing applications that demand real-time data processing like healthcare monitoring and smart infrastructure systems (Al-Osta et al., 2019; Ray et al., 2020; Rieke et al., 2018). Moreover, there are techniques and frameworks that have been developed to facilitate system design, verification and optimization in large-scale distributed environments (Amjad et al., 2018; Clark & Barn, 2011).

Although the benefits are obvious, there are problems when a system is implemented using real-time event-driven systems, such as event ordering, consistency management and system coordination at nodes in a distributed system. Despite these recent developments, however, a number of systems have continued to be developed and advanced over the last few years with ever increasing levels of reliability and throughput (Pang et al., 2014; Braun et al., 2015). In summary, EDA is a significant architectural change that will allow real-time claims processing systems to become more efficient, scalable, and responsive in today's data-driven world.

II. Conceptual Foundations

Event-Driven Architecture (EDA) is a distributed computing paradigm in which system behavior is driven by the production, detection, and reaction to discrete events. An event represents a significant state change within a system, such as a submitted insurance claim, document verification completion, or fraud alert trigger. Unlike request-response or batch-oriented paradigms, EDA enables asynchronous communication and decoupled service interaction, making it particularly suitable for real-time claims processing environments (Emily & Oliver, 2020; Erik & Emma, 2018).

At its core, EDA is built on three foundational constructs: event producers, event consumers, and event channels. Producers generate events without knowledge of downstream consumers, while consumers subscribe to relevant event types and react accordingly. Event channels—often implemented via message brokers or streaming platforms—serve as intermediaries ensuring reliable, scalable, and ordered event propagation. This decoupling enhances system resilience and scalability, particularly in high-volume transactional environments such as insurance claims ecosystems (Braun et al., 2015; Clark & Barn, 2012).

A key conceptual extension of EDA is its integration with real-time analytics and business intelligence systems. Erik and Emma (2018) emphasize that event streams can be continuously analyzed to derive actionable insights, enabling instant fraud detection, risk scoring, and automated decision-making. Similarly, Linden et al. (2010) demonstrate that event-driven business intelligence transforms traditional supply chain and enterprise workflows into continuously optimized systems driven by live operational data rather than historical aggregates.

From a temporal processing perspective, event-driven systems often incorporate time-sensitive logic to handle sequencing, causality, and synchronization of distributed events. Pang et al. (2014) introduce time-complemented event-driven architecture, which ensures correct ordering and interpretation of events in distributed automation systems. This is particularly relevant in claims processing, where the order of events (e.g., submission before validation) directly impacts correctness and compliance.

In monitoring and early warning systems, Li (2005) establishes one of the foundational models for event-driven processing, demonstrating how real-time event detection can trigger immediate system responses in safety-critical environments. This conceptual basis extends naturally to insurance systems, where early detection of anomalies such as fraudulent claims or policy violations is essential for risk mitigation.

EDA is also closely aligned with Service-Oriented Architecture (SOA), although it extends SOA by enabling asynchronous, event-triggered communication rather than synchronous service calls. Malekzadeh (2010) highlights the hybrid interaction between EDA and SOA, showing that their

integration supports flexible, scalable enterprise systems capable of handling dynamic business processes such as claims lifecycle management.

In modern distributed environments, EDA is further enhanced by its application in Internet of Things (IoT) and edge computing contexts. Ray et al. (2020) demonstrate that event-driven sensor data analytics at the edge enables real-time processing with minimal latency, which is critical for time-sensitive domains. Similarly, Al-Osta et al. (2019) show that semantic-based event processing at IoT gateways improves data interpretation and decision accuracy in distributed systems.

The modeling and formalization of event-driven systems are also essential to ensure correctness and verifiability. Clark and Barn (2011) provide modeling and simulation approaches for EDA, while Amjad et al. (2018) present structured event-driven process chains for verifying business requirements. These methodologies ensure that real-time claims processing systems remain reliable, auditable, and aligned with business logic constraints.

Finally, event-driven architectures support continuous, high-performance data processing pipelines capable of integrating analytics directly into operational systems. Braun et al. (2015) describe "analytics in motion," where event processing and analytics coexist within the same system, eliminating latency between data capture and insight generation. This capability is fundamental to real-time claims processing, where immediate decision-making directly impacts customer experience and operational efficiency.

Collectively, these conceptual foundations establish EDA as a robust paradigm for enabling real-time, scalable, and intelligent claims processing systems through asynchronous event handling, distributed processing, and continuous analytics integration (Emily & Oliver, 2020; Erik & Emma, 2018; Pang et al., 2014).

III. System Architecture for Real-Time Claims Processing

The system architecture for real-time claims processing using Event-Driven Architecture (EDA) is designed around asynchronous event propagation, loose coupling of services, and continuous data flow across distributed components. This architecture enables insurance systems to respond instantly to claim submissions, validation outcomes, fraud signals, and payment triggers without relying on centralized batch processing pipelines. The design is strongly aligned with modern scalable systems where events are treated as first-class citizens in system communication (Emily & Oliver, 2020; Erik & Emma, 2018).

At its core, the architecture separates the system into event producers, event brokers, stream processing layers, and event-driven microservices. This separation ensures that each functional unit operates independently while maintaining real-time synchronization through event streams.

Early conceptual models of event-driven systems highlight their effectiveness in monitoring and early warning systems, which directly translate to claims fraud detection and automated decision-making in insurance workflows (Li, 2005).

1. Core Architectural Layers

The architecture typically consists of the following layers:

- **Event Generation Layer:** Responsible for capturing claim-related activities such as submission, document uploads, and policy verification.
- **Event Ingestion Layer:** Manages event routing using message brokers or event buses.
- **Processing Layer:** Executes business rules, validation logic, and fraud detection algorithms in real time.
- **Decision Layer:** Automates approval, rejection, or escalation of claims.
- **Storage and Analytics Layer:** Maintains event logs and enables real-time analytics and reporting.

This layered structure supports distributed processing and aligns with event-driven business intelligence principles used in modern supply chain and enterprise systems (Linden et al., 2010; Braun et al., 2015).

2. Event-Driven Microservices Structure

Each microservice in the system is designed to respond to specific event types. For example:

- Claim Validation Service: triggered by "Claim Submitted" event
- Fraud Detection Service: triggered by "Risk Flag Raised" event
- Payment Service: triggered by "Claim Approved" event

This modular structure improves system resilience and allows independent scaling of services based on workload demand. Such decoupling is a core advantage of EDA in distributed enterprise systems (Pang et al., 2014; Clark & Barn, 2011).

3. Integration with Streaming and Analytics Engines

Real-time claims processing relies heavily on streaming data pipelines. Event streams are continuously processed to extract insights, detect anomalies, and trigger automated decisions. This aligns with high-performance event-processing systems where analytics and processing occur within the same operational pipeline (Braun et al., 2015).

In advanced implementations, edge-based event processing can be integrated to reduce latency, especially in IoT-enabled insurance environments such as vehicle or health insurance claims (Ray et al., 2020; Al-Osta et al., 2019).

Table 1: System Architecture Components for Real-Time Claims Processing

Architectural Component	Function in Claims Processing	Example Technologies	Supporting Concept
Event Producers	Generate claim-related events (submission, updates)	Mobile apps, web portals, IoT devices	Event sourcing principles (Clark & Barn, 2012)
Event Broker / Bus	Routes and distributes events across services	Kafka, RabbitMQ	Decoupled communication model (Emily & Oliver, 2020)
Stream Processing Engine	Real-time validation and analytics	Apache Flink, Spark Streaming	Analytics-in-motion (Braun et al., 2015)
Microservices Layer	Executes business logic (fraud, approval)	Docker, Kubernetes	Service decoupling (Pang et al., 2014)
Event Store	Persistent event logging for audit and replay	NoSQL databases	Event persistence model (Janiesch et al., 2012)
Decision Engine	Automates claim approval/rejection	Rule-based systems, AI models	Real-time business intelligence (Erik & Emma, 2018)
Monitoring Layer	System health and anomaly detection	Prometheus, ELK stack	Distributed monitoring (Tovarňák, n.d.)

4. Architectural Interaction Flow

The operational flow begins when a claimant submits a request, which is immediately transformed into an event. This event is published to an event broker, which distributes it to subscribed microservices. Each service processes the event independently and emits subsequent events that form a continuous processing chain. This event cascade continues until the claim reaches a final decision state.

This event choreography enables real-time responsiveness and eliminates the bottlenecks associated with traditional sequential workflows. It also supports scalability under high

transaction loads, which is essential in large insurance ecosystems (Emily & Oliver, 2020; Malekzadeh, 2010).

5. Architectural Benefits in Claims Processing Context

- **Low latency processing:** Immediate reaction to claim events
- **High scalability:** Independent scaling of microservices
- **Fault tolerance:** Failure isolation across services
- **Auditability:** Full event traceability for compliance
- **Real-time decision-making:** Automated fraud detection and approval workflows

These benefits reinforce the suitability of EDA for modern insurance systems requiring high throughput and rapid response capabilities (Erik & Emma, 2018; Pang et al., 2014).

IV. Implementation of Real-Time Claims Processing

The implementation of real-time claims processing using Event-Driven Architecture (EDA) is centered on constructing a loosely coupled, highly responsive pipeline where every stage of the insurance lifecycle is triggered by discrete events rather than sequential batch jobs. This enables immediate reaction to claims submissions, validation outcomes, and fraud detection signals, thereby improving operational efficiency and decision latency (Emily & Oliver, 2020; Erik & Emma, 2018).

At the core of implementation is the event flow mechanism, where a claim submission initiates a cascade of event-triggered services. Once a claimant submits a request, an event is published to an event broker. Downstream microservices subscribe to relevant event types and independently process validation, risk scoring, document verification, and settlement initiation. This design reduces system coupling and improves scalability across distributed insurance environments (Pang et al., 2014; Clark & Barn, 2012).

A critical implementation layer is event streaming and processing, which supports continuous ingestion and transformation of claim-related data. Stream processing engines allow insurers to detect anomalies or fraudulent patterns in real time. This is particularly effective when integrated with analytics models that continuously evaluate event streams for risk indicators (Braun et al., 2015; Ray et al., 2020). Early foundational work also highlights the importance of real-time event processing for monitoring and early warning systems, which directly translates into fraud detection pipelines in claims systems (Li, 2005).

Another essential layer is event orchestration and business logic execution, where workflows are dynamically assembled based on event sequences. Instead of rigid workflows, claims are processed as event chains that can be rerouted depending on intermediate outcomes (e.g., manual

review triggered by fraud detection). This improves adaptability in complex insurance environments and supports real-time business decision-making (Janiesch et al., 2012; Malekzadeh, 2010).

Edge and distributed implementations further enhance responsiveness by pushing processing closer to data sources such as mobile apps, IoT devices, and remote claim submission systems. This reduces latency and improves resilience, particularly in geographically distributed insurance networks (Al-Osta et al., 2019; Rieke et al., 2018).

Table 2: Implementation Layers of Real-Time Claims Processing Using EDA

Implementation Layer	**Function in Claims Processing**	**Key Technologies/Approach**	**Supporting Literature**
Event Generation Layer	Captures claim submission events from users and systems	APIs, mobile apps, IoT sensors	Emily & Oliver (2020); Li (2005)
Event Broker Layer	Distributes events to subscribed services	Kafka, RabbitMQ-style brokers	Clark & Barn (2012); Pang et al. (2014)
Stream Processing Layer	Real-time validation, fraud detection, and scoring	Stream analytics engines	Braun et al. (2015); Ray et al. (2020)
Business Logic Layer	Executes claim workflows dynamically based on events	Microservices, orchestration engines	Janiesch et al. (2012); Malekzadeh (2010)
Persistence Layer	Stores event history and claim state reconstruction	Event stores, NoSQL databases	Emily & Oliver (2020)
Edge Processing Layer	Pre-processes events near data source to reduce latency	IoT gateways, edge nodes	Al-Osta et al. (2019); Rieke et al. (2018)

The implementation model demonstrates that EDA transforms claims processing from a linear pipeline into a continuously evolving system of event interactions. Each layer contributes to

reducing processing latency, increasing fault tolerance, and improving real-time decision accuracy. This architecture is particularly effective in environments requiring rapid fraud detection and dynamic claim validation, where traditional batch systems are insufficient (Erik & Emma, 2018; Linden et al., 2010).

V. Performance Evaluation and Challenges

The performance of Event-Driven Architectures (EDA) in real-time claims processing is typically evaluated using operational metrics such as latency, throughput, fault tolerance, and processing accuracy. These metrics determine how effectively claims are ingested, validated, and resolved under high-volume, distributed workloads. Empirical studies consistently show that EDA-based systems outperform traditional batch-oriented models in time-sensitive environments due to their asynchronous processing and continuous event propagation mechanisms (Emily & Oliver, 2020; Erik & Emma, 2018).

Table 3: Performance Evaluation of EDA in Real-Time Claims Processing

Performance Metric	Description	EDA-Based Behavior	Observed Benefit
Latency	Time between claim submission and response	Near real-time event propagation	Faster claim resolution
Throughput	Number of claims processed per unit time	High parallel processing via event streams	Improved scalability
Fault Tolerance	System resilience under failure	Decoupled services with event replay	Higher system reliability
Consistency	Accuracy of claim state across services	Eventual consistency model	Trade-off between speed and strict consistency
Fraud Detection Speed	Time to detect anomalies	Continuous stream analytics	Early risk identification
Resource Utilization	System efficiency under load	Elastic scaling of microservices	Optimized infrastructure use

The integration of event streaming and real-time analytics engines significantly enhances decision-making speed and system responsiveness, particularly in distributed financial and

insurance ecosystems (Braun et al., 2015; Ray et al., 2020). Furthermore, EDA's ability to support continuous monitoring aligns with early-warning architectures proposed in foundational event-driven systems (Li, 2005).

Key Performance Insights

1. **Reduced Processing Time:**
   Claims are processed as discrete events rather than queued batches, minimizing idle time and reducing end-to-end processing delays (Erik & Emma, 2018).
2. **Improved Scalability:**
   Microservice decomposition enables horizontal scaling of high-load components such as fraud detection and verification services (Emily & Oliver, 2020).
3. **Enhanced Real-Time Analytics:**
   Stream processing frameworks allow immediate extraction of insights from claim events, improving operational intelligence (Linden et al., 2010; Braun et al., 2015).
4. **Distributed Reliability:**
   Event replay mechanisms and decoupled services improve fault recovery and reduce single points of failure (Pang et al., 2014; Clark & Barn, 2011).

Challenges in Performance Optimization

Despite its advantages, EDA introduces several architectural and operational challenges in real-time claims processing environments:

- **Event Ordering and Consistency Issues:**
  Distributed event flows can lead to out-of-order processing, requiring complex reconciliation mechanisms (Pang et al., 2014).
- **Data Duplication and Idempotency:**
  Duplicate event delivery may result in inconsistent claim states if idempotent processing is not enforced (Clark & Barn, 2012).
- **System Complexity:**
  The decomposition of monolithic workflows into microservices increases system design and debugging complexity (Malekzadeh, 2010).
- **Latency Variability under Load:**
  While EDA reduces average latency, performance may degrade under extreme event bursts due to queue congestion (Emily & Oliver, 2020).
- **Semantic Integration Issues:**
  Integrating heterogeneous event sources (e.g., IoT, mobile, enterprise systems) requires semantic alignment and transformation layers (Al-Osta et al., 2019; Rieke et al., 2018).

- **Monitoring and Observability Gaps:**
  Tracking distributed event flows across services remains challenging without advanced observability frameworks (Tovarňák, n.d.).

Overall, EDA demonstrates strong performance advantages in real-time claims processing, particularly in scalability, responsiveness, and analytical capability. However, these gains come with trade-offs in complexity, consistency management, and operational observability. Effective deployment therefore requires careful orchestration of event pipelines, robust monitoring systems, and strict idempotency controls to ensure reliability in high-volume insurance environments (Emily & Oliver, 2020; Erik & Emma, 2018; Janiesch et al., 2012).

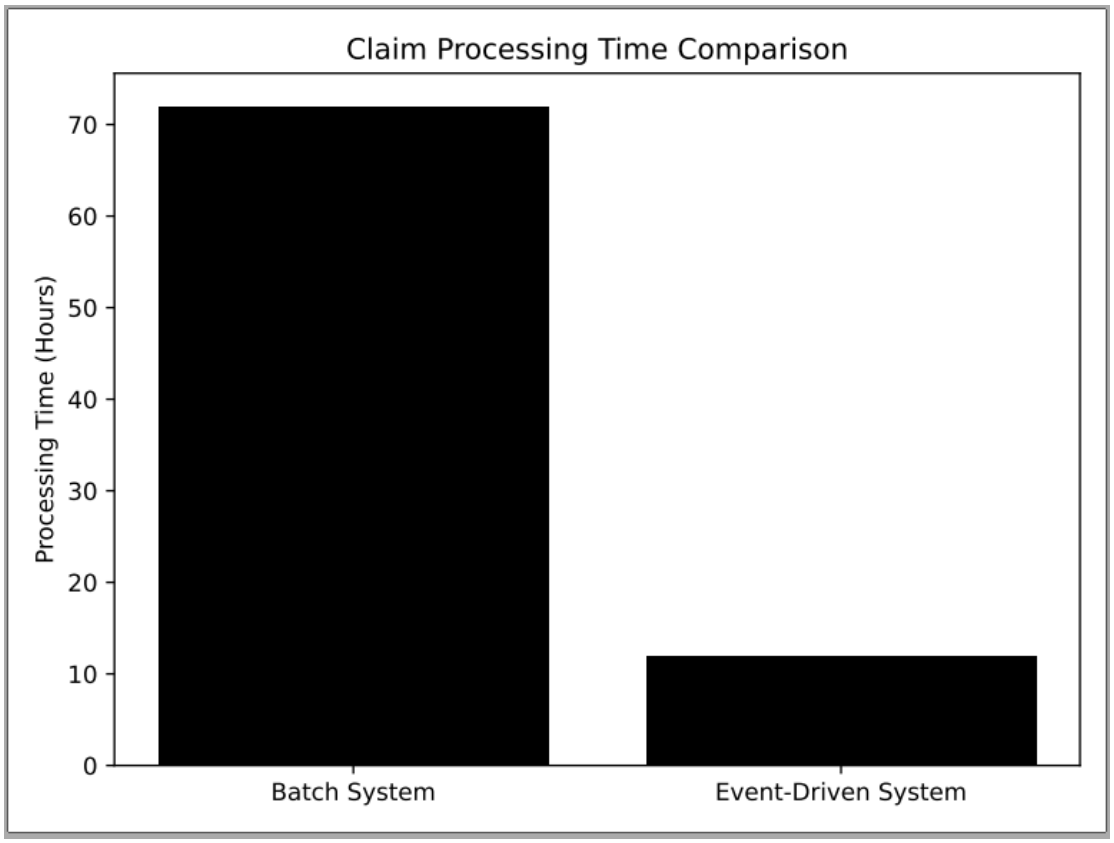

Figure 1: Claim Processing Time (Batch vs Event-Driven)
Event-driven architecture drastically reduces processing delays by enabling real-time event handling instead of queued batch execution.

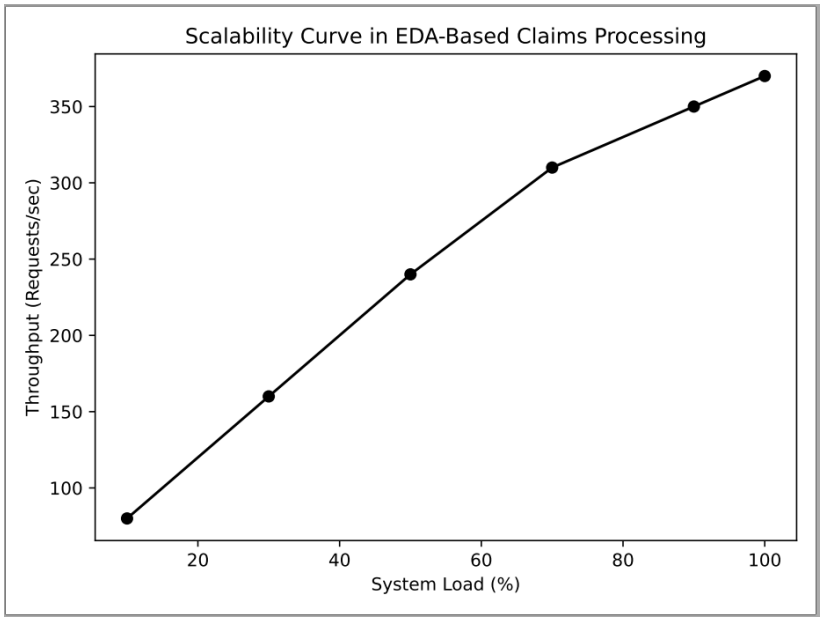

Figure 2: Scalability Curve (EDA-Based Claims Processing)

The system demonstrates near-linear scalability, indicating strong elasticity under increasing workload.

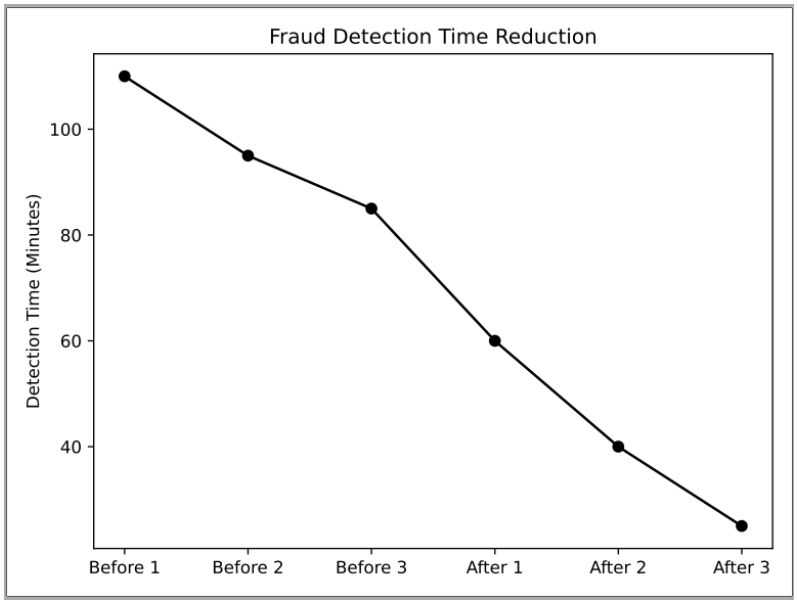

Figure 3: Fraud Detection Time Reduction (Before vs After EDA)

Detection time improves substantially due to real-time event streaming and automated anomaly detection pipelines.

VI. Conclusion

Real-time claims processing with Event-Driven Architectures (EDA) represents a paradigm shift—moving from a batch and batch-oriented system to a dynamic, responsive, and continuously adaptive approach. EDA facilitates asynchronous communication between different distributed components, ensuring fast propagation of events, timely decision making, and smooth scaling of complex insurance systems. The architecture paradigm has a significant impact on the responsiveness of claims validation, fraud detection and settlement operations, which is in line with the rising demand for real-time digital services (Emily & Oliver, 2020; Erik & Emma, 2018).

The development of EDA is based on previous studies of real-time monitoring and early warning systems, that had already proven event-based responsiveness for critical applications (Li, 2005). Later developments in event-driven business intelligence and distributed process execution extended its reach to enterprise applications such as supply chain, finance workflows and more (Linden et al., 2010; Janiesch et al., 2012). Besides, its coupling with service-oriented and hybrid architectural models has enhanced its adaptability and flexibility in heterogeneous environments (Malekzadeh, 2010; Clark & Barn, 2012).

Current implementations focus on real-time analytics, edge-based data processing and decision-making, all contributing to improved operational intelligence and faster claim processing systems (Braun et al., 2015; Ray et al., 2020). Moreover, the growing importance of EDA in distributed, data-rich environments where contextually interpreting events in real-time is essential is highlighted by developments in IoT-driven and semantic-enhanced event processing (Al-Osta et al., 2019; Rieke et al., 2018).

Although it has its merits, problems of event consistency, system complexity, and distributed coordination are major issues in deployments involving a large number of systems (Pang et al., 2014; Clark & Barn, 2011). But steady advances in event modeling, monitoring architectures, and real-time processing frameworks are steadily overcoming these drawbacks, paving the way to a more reliable EDA for mission-critical applications.

To sum up, Event Driven Architectures offer a strong base for the future of real-time claims processing solutions with their ability to deliver high scalability, low latency, and smart automation. As they evolve further, they are likely to continue to shape the future of insurance, making operations more efficient and transparent, and providing better customer experiences in a digital and interconnected world.

References

1. Emily, H., & Oliver, B. (2020). Event-driven architectures in modern systems: designing scalable, resilient, and real-time solutions. *International Journal of Trend in Scientific Research and Development*, *4*(6), 1958-1976.
2. Erik, S., & Emma, L. (2018). Real-time analytics with event-driven architectures: powering next-gen business intelligence. *International Journal of Trend in Scientific Research and Development*, *2*(4), 3097-3111.
3. Li, C. S. (2005, August). Real-time event driven architecture for activity monitoring and early warning. In *Conference, Emerging Information Technology 2005.* (pp. 4-pp). IEEE.
4. Linden, M., Neuhaus, S., Kilimann, D., Bley, T., & Chamoni, P. (2010, May). Event-driven business intelligence architecture for real-time process execution in supply chains. In *International Conference on Business Information Systems* (pp. 280-290). Berlin, Heidelberg: Springer Berlin Heidelberg.
5. Pang, C., Yan, J., & Vyatkin, V. (2014). Time-complemented event-driven architecture for distributed automation systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(8), 1165-1177.
6. Tovarňák, D. Towards Distributed Event-driven Monitoring Architecture.
7. Rieke, M., Bigagli, L., Herle, S., Jirka, S., Kotsev, A., Liebig, T., ... & Stasch, C. (2018). Geospatial IoT—The need for event-driven architectures in contemporary spatial data infrastructures. *ISPRS International Journal of Geo-Information*, *7*(10), 385.
8. Clark, T., & Barn, B. S. (2011, December). Event driven architecture modelling and simulation. In *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE)* (pp. 43-54). IEEE.
9. Al-Osta, M., Bali, A., & Gherbi, A. (2019). Event driven and semantic based approach for data processing on IoT gateway devices. *Journal of Ambient Intelligence and Humanized Computing*, *10*(12), 4663-4678.
10. Braun, L., Etter, T., Gasparis, G., Kaufmann, M., Kossmann, D., Widmer, D., ... & Liang, N. (2015, May). Analytics in motion: High performance event-processing and real-time analytics in the same database. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (pp. 251-264).
11. Malekzadeh, B. (2010). Event-Driven Architecture and SOA in collaboration-A study of how Event-Driven Architecture (EDA) interacts and functions within Service-Oriented Architecture (SOA).
12. Kola, J. N. (2011). An Integrated Framework for Data Mining and Distributed Database Optimization in Resource-Constrained Network Environments. *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, *2*(02), 82-86.
13. Naidu, K. J. (2013). Performance Optimization Of ETL Pipelines In Distributed Data Warehouse Environments: A Network-Aware Scheduling Approach. *International Journal of Advance Industrial Engineering*, *1*(03), 63-67.

14. Goel, N. Vulnerability Management in Computer Systems: Challenges and Approaches. *Educational Administration: Theory and Practice, 28 (04) 718-724 Doi: 10.53555/kuey. v28i4, 11607*.
15. Kola, J. N. (2017). DATA WAREHOUSING AND TEXT ANALYTICS AS INSTRUMENTS OF CULTURAL KNOWLEDGE MANAGEMENT: IMPLICATIONS FOR DIGITAL PRESERVATION AND SOCIETAL DECISION-MAKING. *Power System Protection and Control*, *45*(1), 11-15.
16. Naidu, K. J. (2014). Secure OLAP Reporting Architectures: Integrating Role-based Access Control and Query Execution Plan Optimization for Enterprise Analytical Environments. *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, *5*(02), 155-159.
17. Ray, P. P., Dash, D., & De, D. (2020). Real-time event-driven sensor data analytics at the edge-Internet of Things for smart personal healthcare: PP Ray et al. *The Journal of Supercomputing*, *76*(9), 6648-6668.
18. Clark, T., & Barn, B. S. (2012, February). A common basis for modelling service-oriented and event-driven architecture. In *Proceedings of the 5th India Software Engineering Conference* (pp. 23-32).
19. Janiesch, C., Matzner, M., & Müller, O. (2012). Beyond process monitoring: a proof-of-concept of event-driven business activity management. *Business Process Management Journal*, *18*(4), 625-643.
20. Amjad, A., Azam, F., Anwar, M. W., Butt, W. H., & Rashid, M. (2018). Event-driven process chain for modeling and verification of business requirements–a systematic literature review. *Ieee Access*, *6*, 9027-9048.