Event-Driven Architecture in High-Volume Microservices: A Survey on Industry Practices

Marcus V. S. Silva [1*], Luiz F. C. dos Santos[1†], Fabio Gomes Rocha[1†], Michel S. Soares[1†]

[1*]Computer Science, Federal University of Sergipe, Av. Mal. Cândido Rondon, Rosa Elze, 1861, São Cristóvão, 49100-000, Sergipe, Brazil.

*Corresponding author(s). E-mail(s): marcus.vinicius.3007@gmail.com; Contributing authors: luizfelipecirqueira@outlook.com; gomesrocha@gmail.com; michel@dcomp.ufs.br;
†These authors contributed equally to this work.

Abstract

Event-driven architecture (EDA) has proven itself as a transformative strategy within microservices, recognized for enabling scalable, responsive, and decoupled interactions among system components. This paper draws insights from domains such as e-commerce, healthcare, IoT, and data processing to show how EDA enhances agility and responsiveness, especially under high communication loads and real-time processing demands. We highlight EDA's effectiveness in improving processes like order management, payment handling, and anomaly detection through 13 case studies. These benefits boost efficiency and support better decision-making. Additionally, growing interest in applying EDA to complex systems requiring dynamic adaptation—such as climate risk management and intelligent manufacturing—emphasizes its potential. Still, EDA adoption faces challenges, particularly in managing state, ensuring consistency, and conducting testing, which require deeper analysis.
This paper contributes to ongoing discussions by reviewing the current landscape, challenges, and future directions of EDA, offering a broad perspective on its role in modern software systems. Additionally, one of the goals is to present a comparative analysis between academic perspectives and real-world practices in the technology industry, highlighting where they align and differ in how EDA is understood and applied.

Keywords: Event-Driven Architecture (EDA), Microservices, Scalability, Responsiveness, Decoupling, Real-time processing, State management, Software architecture, Survey

1 Introduction

Event-driven architecture (EDA) has become a key architectural paradigm in the development of distributed and scalable systems, mainly where agility, modularity, and responsiveness are vital to business success. Its capacity to decouple system components and support asynchronous communication makes EDA especially suited for microservices-based applications handling high volumes of data. In sectors such as e-commerce, EDA has demonstrated its value by enabling real-time communication between services related to order management, payment processing, and inventory updates.

While the academic literature offers structured insights into the benefits, challenges, and decision-making processes surrounding the adoption of EDA, there remains a need to understand how these theoretical frameworks align with or diverge from industry practices. A recent systematic literature review by [1] addressed this need by outlining a set of eight key questions regarding the practical application of EDA in microservices, covering areas such as benefits, applicability, cost, challenges, and security.

This article presents a survey designed to complement and contrast those academic findings by capturing the perspectives of professionals actively involved in the design and implementation of EDA-based systems. By gathering 22 industry-based responses to the same set of research questions addressed in the literature review, we aim to provide a comprehensive overview that highlights both alignment and gaps between theoretical recommendations and real-world experiences.

The contributions of this study are threefold: (i) to validate academic insights against practical implementations, (ii) to identify mismatches or blind spots in current literature regarding market realities, and (iii) to offer recommendations for both researchers and practitioners on advancing the use of EDA in high-volume microservice environments.

The remainder of this article is organized as follows: Section 2 presents the methodology, divided into two parts: Section 2.1 describes the Systematic Literature Review (SLR), detailing the databases consulted, inclusion and exclusion criteria, analysis period, and the number of studies reviewed, along with a summary of how the eight research questions were addressed in the literature. Section 2.2 describes the Market Survey, explaining the design of the questionnaire, respondent profiles, the sectors represented, the types of questions used, and the method of collecting answers. This section also emphasizes that the survey's structure mirrored the research questions from the systematic literature review (SLR), enabling a direct comparison between academic and industry perspectives. Section 3 presents and discusses the survey results in light of the literature findings. Section 4 explores the implications of the findings and proposes directions for future research. Section 5 concludes the article with final considerations.

2 Background

Event-driven architecture (EDA) has become a foundational architectural style for building scalable, decoupled, responsive systems. In EDA, components communicate through the emission, detection, and reaction to events, which enables asynchronous and loosely coupled interactions. This paradigm is especially well-suited for microservices architectures, where it supports high flexibility, independent service evolution, and real-time responsiveness[2].

EDA has been successfully applied across diverse domains, including e-commerce, blockchain, healthcare, the Internet of Things (IoT), and data processing, where systems frequently need to handle large volumes of data and support real-time data flow. In e-commerce, for instance, EDA facilitates seamless integration between components that manage orders, payments, and inventory, resulting in an improved customer experience and operational efficiency [3–6]. Similarly, it enables anomaly detection and dynamic process adaptation in healthcare and manufacturing .

Despite these benefits, the adoption of EDA is not without challenges. Common concerns include state management, data consistency, and the testing of asynchronous flows. These technical difficulties underscore the importance of meticulous architectural planning and robust tooling.

From a research perspective, the literature has extensively explored the promises and limitations of exploratory data analysis (EDA). A recent Systematic Literature Review (SLR) [1] distilled the academic discourse into eight guiding questions that address core topics, including applicability, cost, security, and scalability. While these insights form a solid theoretical foundation, questions remain about how well they reflect the realities faced by practitioners in the field. This gap between theory and practice motivates the present study. By surveying professionals involved in designing and operating EDA-based systems, we aim to contrast academic expectations with industry experiences. The goal is to identify alignment and mismatches between scholarly models and real-world implementations, ultimately contributing to a more grounded understanding of EDA's current and future directions.

3 Methodology

The methodology adopted in this study was based on a foundation of thorough research comprising two main phases. First, a systematic literature review was conducted to identify the state of the art regarding the application of Event-Driven Architecture (EDA) in the context of microservices. This review followed a rigorous protocol grounded in well-established guidelines for systematic reviews in software engineering, enabling the collection and analysis of relevant scientific evidence published in recent years. The review provided a comprehensive overview of existing approaches, challenges, and emerging trends related to the adoption of EDA in modern distributed systems, instilling confidence in the study's foundation.

Building upon the findings of the systematic review, a survey was designed and conducted in close collaboration with professionals from the software industry. This collaboration aimed to complement the theoretical insights with empirical evidence from practice, ensuring the study's relevance and applicability. The questionnaire was developed based on the gaps identified in the literature and targeted professionals with proven experience in developing and architecting microservices-based systems. The planning and execution of both phases—the systematic literature review and the industry survey—are detailed in the following sections, including the criteria for inclusion and exclusion, search strategies, questionnaire design, participant demographics, and data collection procedures.

3.1 Systematic literature review

The systematic literature review aimed to characterise, in a structured manner, a guideline for EDA from a broad perspective. A systematic mapping has been chosen as the research instrument to achieve this. A systematic mapping is a comprehensive and rigorous review of a research field or topic, allowing a broad view of the paths taken and their respective gaps [7]. The systematic mapping presented in this article follows the guidelines of Kitchenham and Charters [8] and is divided into three phases: planning, execution, and finally, communicating the results. First, the topic to be researched was defined. Next, the applied methodology and the selection of databases where the articles were searched were detailed. Questions to be answered by this study and the research chain related to the topic were prepared, all using a tool named Parsifal [9].

A detailed protocol was followed to classify the articles, ensuring the transparency and reproducibility of the study. This protocol included the following steps:

- Definition of Inclusion and Exclusion Criteria
- Article selection
- Classification and Analysis of Articles
- Definition of research questions
- Number of publications per year

Articles that addressed EDA, even partially, were included as part of the inclusion criteria. For the exclusion criteria, duplicate articles, abstracts or expanded summaries, non-English articles, those not discussing EDA, and secondary studies were removed, as shown in Table 1.

Table 1 Criteria

Inclusion	Articles that deal with EDA
Exclusion	Duplicate articles Articles published as abstracts or expanded summaries Articles that are not written in English Articles that do not discuss EDA Secondary studies

A search was conducted using the terms `(‘‘eda’’ OR ‘‘event-driven architecture’’ OR ‘‘eventual architecture’’) AND (‘‘microservices’’ OR ‘‘faas’’ OR ‘‘function as a service’’ OR ‘‘msa’’ OR ‘‘nanoservice’’ OR ‘‘nanoservices’’)`, without time restrictions, in the ACM, IEEE, Web of Science, ScienceDirect, and Scopus databases on August 3, 2024. These selected databases are hybrid, serving as both search engines and bibliometric databases, and are widely adopted in studies to ensure comprehensive coverage for systematic reviews, as noted

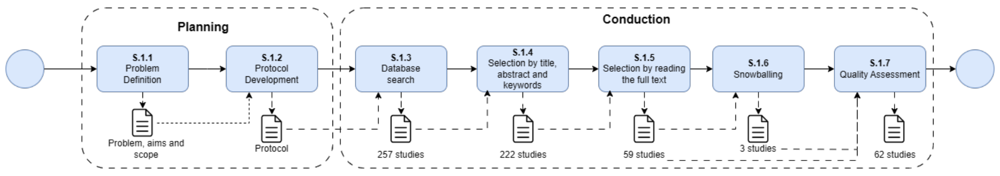


Fig. 1 Selection of Articles

in [10]. A total of 257 relevant articles were identified that addressed, even partially, the concept of Event-Driven Architecture.

After removing duplicates and applying the inclusion and exclusion criteria, 47 duplicate articles were eliminated, resulting in 222 articles for verification. Of these, 62 were considered relevant, as they addressed event-driven architecture in their titles and abstracts, meeting the study requirements. Meanwhile, the remaining 160 were excluded for failing to meet the established criteria, as shown in Figure 1. In the stages of this study, articles that met the inclusion and exclusion criteria were analyzed to find answers to the questions, using structured questions that can guide future research sequences and evaluate the research process [11], as described in Table 1.

The annual increase between 2019-2022 in the number of articles on event-driven architecture, reflecting this approach's growing interest and adoption. These publication trends provide insights into the evolution and maturity of event-driven architecture practices, highlighting the development of academic and practical interest in this topic. This information is essential as it allows the identification of research trends, the measurement of the growth in adopting this technology and the understanding of periods of significant development and innovation. Furthermore, it helps recognize the increasing relevance of the topic in the developer and researcher communities, providing a foundation for future investigations and improvements in the field.

Table 2 Research Questions and Justifications

Question Number	**Question Text**	**Justification**
Q1	What are the benefits of using the Event-Driven Architecture model?	To assess the immediate benefits that this architectural model brings.
Q2	How to identify if Event-Driven Architecture can be used in your solution?	To evaluate possible scenarios already known by the community and assist in decision-making.
Q3	What is the impact of using this architectural model in an unsuitable context?	To identify possible impacts of incorrect decision-making.
Q4	Do engineers have a guideline for selecting Event-Driven Architecture today?	To assess best practices and strategies to optimize the development process.
Q5	What are the main challenges when using this architectural model?	To demonstrate added value to stakeholders.
Q6	In terms of cost, does this architectural model have any significant impact?	To identify possible factors that increase the cost of this architectural model.
Q7	What are the challenges and solutions to ensure data security and privacy in Event-Driven Architecture-based systems?	To identify challenges in implementing this architectural model.
Q8	In which scenario was Event-Driven Architecture used?	To identify contexts and scenarios where this architectural model has been applied.

Table 3 Articles Per Source

Source	Percentage
ACM Digital Library	49%
Web Of Science	14%
Scopus	18%
Science Direct	9%
IEEE	10%

3.1.1 RQ1. What are the benefits of using Event-Driven Architecture?

EDA offers several benefits for building distributed systems and microservices, promoting scalability, modularity, and flexibility in various contexts. One of the main benefits of EDA is scalability, allowing services to scale independently according to demand [12] and facilitating the efficient processing of large volumes of data [13]. Modularity and decoupling enable new services to be integrated into the system without significantly impacting the existing environment [12].

The flexibility of EDA is reflected in the possibility of using different technology stacks for each service [14], as well as the independence of updates, allowing each service to be updated independently without affecting the entire system [15].

Therefore, adopting EDA facilitates the creation of highly scalable, flexible, and robust systems with lower maintenance complexity and the ability to handle large volumes of real-time data.

Table 4 Benefits of Event-Driven Architecture (EDA)

Benefit	Description	Reference
Scalability	EDA allows for the independent scalability of services, enabling each microservice to be adjusted according to demand.	[16], [17], [18], [19], [20], [21], [22].
Decoupling	Facilitates decoupling between services, promoting a modular architecture that simplifies system maintenance and evolution.	[23], [24], [25], [17], [20], [26]
Flexibility	Supports multiple technology stacks in different services, allowing updates and changes without impacting the entire system.	[24], [10], [27], [28], [29], [17], [20]
Reactivity	Facilitates real-time response to events, improving the system's ability to react quickly to environmental changes.	[30], [31], [32], [33]
Resilience	The architecture continues operating even with individual microservices failures, increasing fault tolerance and system robustness.	[34], [35], [36], [37], [38]

3.1.2 RQ2. How to identify if one can use Event-Driven Architecture?

To determine if Event-Driven Architecture (EDA) is suitable for a solution, it is essential to assess the specific requirements and characteristics of the system in question. EDA is often employed when there is a need for reactivity to events, distributed processing, and interactions between different system components. One study suggests that systems requiring real-time response to events and strict event-based data processing requirements are ideal candidates for EDA [12]. Similarly, systems facing performance bottlenecks in specific modules can benefit from migrating to an event-driven architecture [14].

EDA also excels in solutions that require distributed processing and interactions among multiple components, ensuring that communication between them is efficient and scalable [39]. In applications demanding low latency, parallel execution, and real-time responses to large volumes of data, EDA can be successfully utilized to optimize microservices, as shown in the application of EDA for OLDI microservices with very low latencies [40].

Finally, in big data solutions, such as processing large volumes of data generated by autonomous vehicles, EDA provides a practical framework for managing real-time data streams [18].

3.1.3 RQ3. What is the impact of using EDA in an unsuitable context?

When applying the EDA in unsuitable contexts, several negative impacts can arise, such as increased complexity, maintenance difficulties, and performance inefficiencies. First, when EDA is used in systems that do not require high scalability or in scenarios where asynchronous communication is not essential, it can introduce unnecessary challenges. For instance, [14] points out that, in small systems, using EDA can generate unnecessary overhead, particularly regarding communication Between services and network latency, [12] emphasizes that coordinating microservices can lead to data consistency issues and challenges in synchronization and validation across services, further increasing system complexity.

Finally, implementing EDA in unsuitable contexts can directly impact the scalability and security of the system. According to [41], in contexts where events are not predominant or synchronous communication is preferred, EDA can increase latency, hinder maintenance, and result in more significant infrastructure costs without providing proportional benefits to the system. According to [16], applying EDA in scenarios where events are not central to the system requirements can lead to scalability difficulties and increased operational costs.

3.1.4 RQ4. Do engineers have any guidelines for selecting Event-Driven Architecture today?

According to [14], engineers should consider systems that require scalability, component autonomy, and real-time response as potential candidates for implementing EDA. Additionally, [39] highlights the need for scalability, parallel execution, and asynchronous communication as determining factors for using EDA.

On the other hand, [13] mentions that EDA is particularly useful in systems dealing with large volumes of real-time data, such as in extreme weather events. Additionally, [42] reinforces the use of EDA in scenarios involving high scalability, real-time event processing, and service decoupling.

Moreover,[33] suggests that using a broker, such as RabbitMQ or Kafka, can be an effective strategy, especially in microservice architectures requiring high availability and parallel processing.

3.1.5 RQ5. What are the main challenges of using this architectural model?

One of the main challenges is synchronization and validation between microservices, where data replication and fault tolerance are critical issues is cited [43]. The need to ensure that event-based constraints are correctly applied is one of the points highlighted by [12]. The difficulty of dealing with latency and the communication overhead between services is also a recurring problem, as described by [14].

Another challenge is state management, which requires ensuring data consistency across multiple services, particularly in distributed scenarios [23]. Additionally, event logging and tracking across different services can be complicated due to the asynchronous nature of EDA, as indicated by [44].

Finally, the challenge of ensuring data security in transit, especially in distributed environments, is emphasized in [45].

3.1.6 RQ6. In terms of cost, does this architectural model have any significant impact?

EDA significantly impacts distributed systems' operational and implementation costs. First, the complexity of synchronization, validation, and data replication between microservices can increase operational costs, requiring careful management to avoid failures and ensure proper system recovery [46]. Additionally, migrating applications from a monolithic framework to microservices can incur extra costs in managing communication and latency between microservices and maintaining multiple instances.

Operational costs may be increased by the need for dynamic adjustments in systems using EDA, requiring an adaptable thread model and the ability to respond quickly to changes in workload. In scenarios where scalability is crucial, such as applications requiring high-demand processing, EDA may lead to higher initial costs due to the necessary infrastructure. Still, it can also provide long-term savings by enabling more efficient reactivity and scalability [29].

In conclusion, while Event-Driven Architecture may initially raise costs due to its complexity and the need for robust infrastructure, scalability, efficiency, and reactivity benefits can justify these long-term investments.

3.1.7 RQ7. What are the challenges and solutions for ensuring data security and privacy in Event-Driven Architecture-based systems?

EDA presents several challenges concerning data security and privacy, mainly due to its distributed nature and asynchronous communication between microservices. The main challenges identified include the transmission of sensitive data, the need for robust authentication and authorization, and protection against malicious events. Data transmission frequently occurs between different microservices, increasing the risk of vulnerabilities, as it is crucial to ensure that unauthorized events do not compromise the integrity of services [42].

Various solutions have been proposed in the literature to address these challenges. Implementing authentication, authorization, encryption, and proper access control mechanisms is essential for ensuring data security [12]. Encryption to protect data in transit, coupled with continuous monitoring and event auditing, is considered a best practice for detecting suspicious behaviour and mitigating risks [41].

In summary, security and privacy in EDA-based systems require a comprehensive approach that combines robust security mechanisms with the implementation of continuous monitoring and auditing practices, thus ensuring the integrity and protection of data throughout the entire lifecycle of the services.

3.1.8 RQ8. In what scenarios has Event-Driven Architecture been used?

EDA has been widely applied in various scenarios, demonstrating its flexibility and effectiveness in distributed systems. One notable example is the e-commerce context, where EDA facilitates interaction between different microservices to process orders, payments, and inventory updates asynchronously enabling systems to respond quickly to dynamic events, improving user experience and operational efficiency.

Beyond e-commerce, EDA has been utilized in scenarios optimizing latency for OLDI microservices applications, where quick response is critical for maintaining scalability under high processing loads [40] or logistics services how demonstrated [47]. In IoT systems, EDA enables real-time responses to events by promoting an architecture that adapts to rapid changes in the environment [44].

In the financial context, EDA is applied in payment services and transaction settlement, demonstrating its effectiveness in real-time information processing [31].

3.2 Industry Survey

To complement the findings of the Systematic Literature Review (SLR) and ensure a holistic understanding of Event-Driven Architecture (EDA), a market survey was conducted under the title ”Refining Inquiry: Validating Research Questions on Event-Driven Architecture (EDA).” The primary goal of this survey was to validate the relevance and applicability of the research questions raised in the academic context by comparing them with the experiences and insights of industry professionals.

Participants and Profiles The survey collected responses from 22 professionals actively involved in the design and implementation of software systems. Participants were primarily recruited through professional social networks, such as LinkedIn, as well as through direct sharing within specialized communities and companies operating in various sectors of the technology industry.

Notable companies represented among the respondents include AssureSoft, Dev.Pro, Banco Carrefour, XP Investimentos, Stone, Pagcerto, Neon, Nubank, and iFood. These organizations span various industries, including fintech, software consultancy, retail banking, and technology platforms.

The professional profiles of the participants were as follows:

Table 5 Profile of Survey Participants

Professional Role	**Percentage**
Solution Architects	27.3%
Senior Developers or Engineers	50%
Software Architects	22.7%

This distribution reflects a strong technical background and hands-on experience with modern architectural patterns, including EDA, thereby reinforcing the validity and applicability of the collected data.

3.2.1 Survey Structure

The structure of the survey was carefully designed to mirror the eight research questions (RQs) addressed in the SLR. This alignment ensures a direct comparison between academic insight and professional practice. For each research question, respondents were asked using the following:

Open-ended questions to collect qualitative insights, personal experiences, and contextual observations.

This format approach allowed quantitative validation and qualitative enrichment of the academic findings. The consistent question structure between the SLR and the market survey facilitates the identification of convergences and divergences between the two knowledge domains.

3.2.2 Objectives and Relevance

The objective of the survey is to bridge the gap between academic research, which typically emphasizes theoretical rigor and structured methodologies, and industry practice, which focuses on practicality, cost-effectiveness, scalability, and security. By refining the research questions based on feedback from professionals, the study aims to ensure that the proposed insights are not only academically sound but also highly relevant and actionable in real-world scenarios.

4 Results

This section presents the results obtained from both the systematic literature review and the industry survey, highlighting key insights and patterns identified in the study. The findings are organized to contrast and compare the perspectives of academia and industry on the adoption and implications of Event-Driven Architecture (EDA) in microservices-based systems. The analysis, which is comprehensive in nature, focuses on identifying points of convergence and divergence, particularly in terms of perceived benefits, challenges, and practical considerations. By juxtaposing theoretical and empirical data, this section aims to provide a comprehensive understanding of how EDA is conceptualized and applied in real-world scenarios, as opposed to academic discourse.

4.0.1 Alignment and Divergences Between Industry and Academic Perspectives on Benefits

Event-Driven Architecture (EDA) offers a wide range of benefits, recognized by both the industry and the academic community. However, the emphasis placed on each benefit varies depending on the perspective.

From the **Industry perspective**, the most cited advantages are *decoupling* of components, *scalability* to handle large volumes flexibly, *resilience* to failures, *real-time processing*, and overall system *flexibility* and *agility*. Professionals often highlight the practical operational gains, such as easier troubleshooting (e.g., through Dead Letter Queues), clear context boundaries (Bounded Contexts), cost reduction, and better responsiveness in integrations and updates.

On the other hand, the **academic perspective** focuses more systematically on architectural aspects such as *scalability*, *decoupling*, *flexibility*, *reactivity* (real-time response to changes), and *resilience*. Academic works emphasize how EDA enables modularity, facilitates the independent evolution of services, improves the robustness of the system, and supports heterogeneous technological environments.

Table 6 Industry Perspective vs Academic Perspective

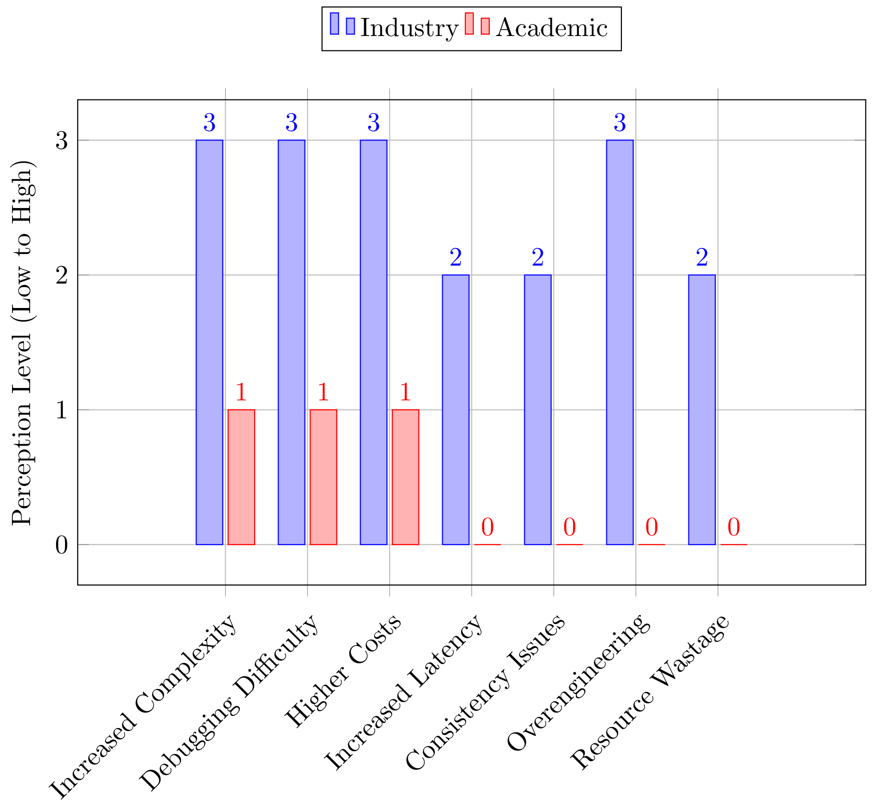
Benefit	**Industry Perspective**	**Academic Perspective**
Scalability	Seen as one of the most important benefits to handle high volumes and growth flexibly.	Independent scalability of services to meet varying demands. [16], [17], [18]
Decoupling	Strongly emphasized to allow independent evolution, flexibility, and resilience between services.	Promotes modularity, simplifies maintenance and facilitates independent service updates. [23], [24]
Flexibility	Allows adding or updating services easily, adapting to business needs and promoting responsiveness.	Supports use of multiple technology stacks and independent updates without affecting the whole system. [24], [10]
Resilience/Fault Tolerance	Important for ensuring systems can recover and maintain operation after failures. Retry policies and dead-letter queues are cited.	Failures are localized, system remains operational, increasing robustness and fault tolerance. [34], [35]
Real-Time Processing/Responsiveness	Emphasized for responsiveness, high integration speed, and better user experiences.	Facilitates real-time reactions to environmental changes and enhances throughput. [30], [31]
Auditability/Traceability	Highlighted in comments about reconstructing entity states, observability, and system troubleshooting.	Events provide an immutable log that improves auditability and traceability. [30]
Extensibility	New services and consumers can be added without impacting producers, allowing system growth.	Enables system evolution by easily integrating new components without major changes. [24]

Table 7 Overlap and Distinctions in EDA Benefits

Situation	**Number of Benefits**	**Benefits**
Found in both Industry and Academic	7	Scalability, Decoupling, Flexibility, Resilience, Real-Time Processing, Auditability, Extensibility
Found only in the Industry	8	Asynchronicity, Easier Troubleshooting, Clear Context Boundaries, Cost Avoidance, High Responsiveness, Ease of Integration, Component Reuse, Agility
Found only in Academic	0	None

4.0.2 Perception of Risks When Using EDA in Unsuitable Contexts: A Comparison Between Industry and Academic

While Event-Driven Architecture (EDA) is widely recognized for enabling scalable, reactive, and distributed systems, its misuse in unsuitable contexts can introduce significant challenges. Industry perspectives highlight a range of practical issues, such as increased complexity, higher operational and infrastructure costs, difficulty in debugging and testing, latency overhead, and risks of data inconsistency. These challenges often arise when EDA is applied to systems that do not inherently require asynchronous processing or event-based coordination, leading to overengineering and maintainability problems. In contrast, the academic literature generally frames EDA within ideal conditions, emphasizing its benefits when applied to systems with strict real-time requirements, distributed environments, and high data throughput demands. Studies rarely address the negative implications of misuse directly, often assuming the presence of appropriate technical maturity and use-case alignment. This discrepancy underscores the importance of careful architectural evaluation prior to adopting EDA, especially in simpler systems where the trade-offs may outweigh the potential advantages.



4.0.3 How to identify if one can use Event-Driven Architecture?

Industry Perspective

Identifying whether Event-Driven Architecture (EDA) is the right fit for a system starts with analyzing **business needs**, **system requirements**, and the company's **future vision**.

EDA is particularly recommended in scenarios such as:

- When *asynchronous communication* is required, allowing processes to avoid blocking while waiting for responses.
- When the solution demands *independent scaling* of services.
- When *decentralized workflows* or *dynamic business rules* are needed.
- When handling *large volumes of data*, *real-time monitoring*, *resource tracking* (e.g., databases, virtual machines), or *financial transactions.*
- When *high scalability*, *loose coupling*, and *service independence* are critical.
- When *event traceability* is necessary to monitor state changes.
- When business domains present *complex workflows* or *multiple consumers* for the same event.
- When solutions require *real-time reactions* to system events and *parallel processing.*
- When teams have expertise in *asynchronous messaging*, *distributed systems*, and *event modeling.*

It is important to note that EDA may not be the best choice when the system requires **immediate response times**, **strong consistency**, or when the solution is **simple** and does not justify the added complexity and operational cost.

Industry professionals often rely on the following checklist to assess the suitability of EDA:

- **High decoupling need**: Services must evolve independently.
- **Tolerance for asynchrony**: The business can handle eventual consistency.
- **Complex workflows**: Processes are better modeled through events.
- **Scalability demands**: The system must manage variable loads efficiently.
- **Audit and logging requirements**: Event traceability is crucial.

Academic Perspective

From an academic standpoint, the decision to adopt EDA depends on an assessment of the system's requirements and characteristics.

Research highlights that EDA is particularly suitable when:

- There is a *need for real-time event reaction* and *strict event-based data processing* [12].
- Systems experience *performance bottlenecks* that can be mitigated through distributed, decoupled event processing [14].
- *Distributed systems* require *efficient and scalable communication* between components [39].
- Applications demand *low latency*, *parallel execution*, and *real-time responses* to large data volumes, such as in *microservices optimization* with extreme latency sensitivity [40].

Academic studies reinforce industry practice: EDA is an excellent choice for building **highly reactive, scalable, and decoupled** systems, provided that the added complexity is justified by the business and technical gains.

4.0.4 Do engineers have any guidelines for selecting Event-Driven Architecture today?

Academic Perspective

Academic research suggests that there are clear indicators for selecting Event-Driven Architecture (EDA). According to [14], systems requiring high scalability, autonomy between components, and real-time responsiveness are strong candidates for EDA. Similarly, [39] emphasizes the importance of scalability, parallel execution, and asynchronous communication as decisive criteria. Other studies, such as [13], reinforce the application of EDA in handling large volumes of real-time data (e.g., monitoring extreme weather events), while [42] points out its suitability for highly scalable, event-driven, and decoupled systems. Finally, [33] suggests the strategic use of brokers like RabbitMQ and Kafka, particularly in microservices architectures that require high availability and efficient parallel processing.

Overall, the academic vision provides a consistent set of guidelines based on system requirements, mainly focusing on technical needs like performance, scalability, and real-time data handling.

Industry Perspective

From the industry point of view, the presence of formal guidelines is more varied. While many professionals acknowledge that there are solid design patterns, cloud-native solutions, and case studies supporting the adoption of EDA, the decision-making process often leans heavily on engineering judgment and accumulated experience rather than strict formal frameworks.

Some companies apply structured approaches like CBAM (Cost Benefit Analysis Method) and ATAM (Architecture Tradeoff Analysis Method) to validate EDA adoption. Others emphasize the importance of having proper infrastructure in place—such as event brokers, governance models, error handling strategies, and observability tools—before proceeding.

Common criteria mentioned by industry practitioners include:

- Real-time processing needs.
- Scalability requirements.
- Loose coupling between services.
- Data/event volume forecasts.
- Complexity versus cost trade-offs.

Additionally, some caution that despite the abundance of material (articles, tutorials, case studies), without disciplined event schema governance, monitoring, and operational tooling, adopting EDA can lead to architectural chaos.

Thus, although the industry has access to guidelines and frameworks, EDA adoption often remains driven by context-specific needs, team expertise, and business fit.

Table 8 Comparison of Academic and Industry Perspectives on EDA Selection Guidelines

Aspect	**Academic Perspective**	**Industry Perspective**
Existence of Guidelines	Strong guidelines based on system characteristics (scalability, autonomy, real-time needs)	Partial presence: experience, decision records, and architectural methods (e.g., CBAM, ATAM) guide decisions
Criteria	Scalability, autonomy, real-time data handling, distributed systems efficiency	Real-time needs, scalability, loose coupling, fault tolerance, cost-benefit feasibility
Formal Methods	Strong reliance on theoretical models and proven studies	Some formal methods (CBAM, ATAM), but mainly practical and experience-driven
Concerns	Complexity must be justified by benefits	Complexity, operational governance (observability, event versioning)
Tools and Frameworks	Recommended use of brokers (Kafka, RabbitMQ) in complex systems	Emphasis on proper infrastructure, monitoring, and discipline for EDA adoption

4.0.5 What are the main challenges of using this architectural model?

Industry Perspective

In the industry, the primary challenges revolve around the practical aspects of implementation, such as eventual consistency, complexity of event management, debugging, and monitoring. The complexity of guaranteeing the order of events and maintaining data consistency in distributed systems is often highlighted. Event duplication and idempotency, as well as managing infrastructure (e.g., message brokers, API gateways), are also significant issues. Industry emphasizes the increased operational overhead and the difficulties in ensuring a seamless event flow across decoupled services. Furthermore, debugging and tracing events across multiple services in an asynchronous environment pose substantial difficulties. Observability and testing are highlighted as critical areas where proper tools and frameworks are necessary for efficient maintenance and support.

Academic Perspective

While acknowledging the practical concerns, emphasizes theoretical challenges such as synchronization and validation between microservices, data replication, and fault tolerance. The need for proper state management across distributed systems is another key issue. In particular, academic discussions focus on ensuring data consistency through eventual consistency mechanisms and dealing with the latency and communication overhead between services. Event logging and tracking across services are complicated by the asynchronous nature of EDA, making it challenging to trace event flows and ensure integrity. Event schema evolution and the application of event-based constraints are additional focal points in academic research.

Despite the differences in focus, both sectors agree on the complexity of implementing EDA, the difficulty in maintaining consistency, and the challenges related to asynchronous communication and debugging.

Table 9 Comparison of Industry and Academic Perspectives on EDA Challenges

Aspect	**Industry Perspective**	**Academic Perspective**
Eventual Consistency	Difficult to manage data consistency in distributed systems, especially with asynchronous communication. Operational overhead and event duplication are key concerns.	Focus on ensuring eventual consistency across services. Data replication and fault tolerance are critical challenges.
Event Ordering	Ensuring the correct order of events is complex, especially when dealing with many events in a distributed system.	Synchronizing event order across distributed services is a recurring problem.
Event Duplication	Handling duplicate events and ensuring idempotency to avoid data errors. Requires careful design to handle duplicates safely.	Event duplication is an inherent challenge in EDA and requires strategies to handle it effectively.
Infrastructure Management	Managing infrastructure, including message brokers, APIs, and observability tools, is crucial for ensuring smooth event processing.	The complexity of managing distributed infrastructure (e.g., brokers, queues) is a major challenge, especially when dealing with high traffic and event volume.
Observability	Debugging and tracing events across services is tough. A centralized logging system can help, but is not always implemented. Monitoring tools are crucial.	Observability is hard due to the distributed nature of EDA. Event tracking across services is challenging and often requires sophisticated tools.
Testing	Testing asynchronous, distributed flows is difficult. Multiple components must be running locally to simulate real behavior.	Testing the interaction of services in an asynchronous environment is complex. Simulating failures and event flows is hard.
Latency	Managing latency is a concern, especially in real-time systems. It affects event processing and the speed of feedback.	Latency in communication between services is a significant issue, affecting system performance and user experience.
State Management	Ensuring data consistency and managing state across multiple distributed services is a key challenge.	State management is a critical challenge in ensuring data consistency across microservices, particularly in distributed systems.
Event Schema Evolution	Evolving event schemas without breaking backward compatibility is a concern, especially with many consumers.	Schema evolution requires careful management to ensure backward compatibility without disrupting the system's operation.
Event Logging	Logging and tracing event flows across services is a significant challenge, making troubleshooting difficult.	Event logging and tracking across distributed services is complicated by the asynchronous nature of EDA.
Error Handling	Error detection is more difficult due to asynchronous failure handling and dead-letter queues.	Error handling is difficult in asynchronous environments, where failures are often non-local and not immediately detectable.

4.0.6 In terms of cost, does this architectural model have any significant impact?

Industry Perspective

Event-Driven Architecture (EDA) can significantly impact costs, both positively and negatively. Initially, infrastructure costs tend to be higher due to the need for additional tools, such as message brokers and monitoring systems. Moreover, the architecture requires a highly skilled team for maintenance and operation, which can drive up salary costs.

On the other hand, EDA can provide savings in highly scalable scenarios. The ability to scale individual components independently reduces the need to scale monolithic systems, potentially resulting in significant savings. Additionally, service decoupling and asynchronous processing reduce idle time and improve efficiency, allowing processes to be run on cheaper services.

However, EDA also requires initial investments in event management tools, along with additional costs for team training. Long-term, maintaining the architecture and adding new functionalities may become easier, leading to savings by reducing system complexity.

Academic Perspective

In Academy, EDA is viewed as an architecture that can increase operational costs due to the complexity of synchronization, validation, and data replication between microservices. Furthermore, migrating applications from monolithic frameworks to microservices can incur additional costs related to communication and latency between services. The need for a dynamic thread model and the ability to respond quickly to workload changes can further increase operational costs.

Nonetheless, EDA can offer long-term savings, especially in scenarios requiring high scalability. The architecture allows for better responsiveness to workloads, potentially leading to greater resource efficiency, despite the higher initial infrastructure costs.

4.0.7 What are the challenges and solutions for ensuring data security and privacy in Event-Driven Architecture-based systems?

Data security and privacy in Event-Driven Architecture (EDA) present several challenges and solutions. The challenges include unauthorized access, data leakage, event manipulation, and ensuring compliance with regulations such as GDPR and HIPAA. These challenges arise due to the distributed and asynchronous nature of EDA-based systems, which increase the attack surface for potential security breaches.

Industry Perspective

The industry emphasizes the need for robust security measures like encryption (both in transit and at rest), access control (e.g., role-based access control or RBAC), and audit logging to ensure data privacy and prevent unauthorized access. Furthermore, the distributed nature of EDA makes it harder to control access and ensure data security across multiple services. Many companies use internal message brokers, but even in such environments, it is crucial to ensure data confidentiality through secure communication protocols like TLS and proper authentication mechanisms (e.g., JWT, mTLS).

Academic Perspective

From an academic standpoint, the challenges are primarily linked to the transmission of sensitive data across various microservices and the complexity of maintaining secure, asynchronous communication channels. Solutions such as encryption, continuous monitoring, event auditing, and proper authentication and authorization mechanisms are proposed to ensure the security of data. Moreover, a decentralized approach to control access and avoid data manipulation is essential for maintaining integrity throughout the system. The academic literature emphasizes the need for both strong security measures and ongoing auditing to protect data during transit and in storage.

Aspect	**Industry Perspective**	**Academic Perspective**
Data Transmission	Sensitive data in transit must be encrypted using TLS, secure brokers, and strong authentication mechanisms.	Encryption in transit and continuous monitoring are essential for ensuring security in distributed systems.
Access Control	Role-based access control (RBAC) and zero-trust communication help control which components can access the events.	Robust authentication, authorization, and fine-grained access control are necessary for securing microservices.
Event Integrity	Protecting against event manipulation requires solutions like digital signatures and Message Authentication Codes (MACs).	The need to protect against malicious events highlights the importance of ensuring event integrity and implementing validation mechanisms.
Data Leakage	Solutions to prevent data leakage include event masking, anonymization, and proper event storage policies.	Ensuring data privacy requires encryption, data minimization, and ensuring that sensitive data is not included in events.
Compliance and Regulations	Regulatory compliance (GDPR, HIPAA) is addressed using data governance frameworks and audit logs.	Academic approaches emphasize the importance of auditing and compliance measures for privacy regulations, particularly for sensitive data.
Monitoring and Auditing	Continuous monitoring, logging, and auditing help mitigate risks and detect suspicious activity.	Ongoing monitoring, event auditing, and traceability are critical for detecting anomalies and ensuring data protection.
Data Retention	Policies for data retention ensure that unnecessary data is deleted, preventing storage issues.	Data retention policies and automated deletion of old events are necessary to avoid over-retention and comply with privacy regulations.

Table 10 Security and Privacy Challenges and Solutions in EDA Systems: Industry vs. Academic Perspectives

4.0.8 In what scenarios has Event-Driven Architecture been used?

Industry Perspective

Event-Driven Architecture (EDA) has been extensively used in various real-time systems, including:

- **Financial Systems:** Real-time transaction processing, fraud detection, payment processing, and integration with external systems like SAP and IRS APIs. EDA is ideal for handling high-volume transactions with low failure margins.
- **E-commerce Platforms:** Order processing, inventory management, payment confirmation, and shipping. EDA allows decoupling of services, enabling scalability and efficient handling of high traffic loads.
- **IoT Systems:** Device communication and real-time data processing. EDA enables responsive, scalable architectures to handle telemetry data from various devices.
- **Social Media:** Event-driven systems handle user actions like posting, liking, and sharing, triggering various backend processes such as feed updates, analytics, and recommendation engines.
- **Healthcare:** Event streams from patient monitoring devices, triggering alerts, updates, and automating responses to changing conditions.
- **Supply Chain and Logistics:** Inventory updates, shipment tracking, and procurement flows.
- **Real-Time Analytics and Reporting:** Event-based processing allows real-time updates in analytical systems, improving responsiveness and decision-making.
- **Ticket Management Systems:** EDA is used to update the state of tickets across multiple services without waiting for responses, ensuring a decoupled and efficient processing workflow.

Academia Perspective

In the academic field, Event-Driven Architecture has demonstrated its flexibility and effectiveness in several key scenarios:

- **E-commerce:** EDA is crucial for handling asynchronous interactions between microservices for processing orders, payments, and inventory updates, which enhances user experience and operational efficiency.
- **Microservices and Low-Latency Applications:** EDA is widely used to optimize latency in applications that require quick responses and scalability under high processing loads, such as microservices-based platforms.
- **Logistics and Supply Chain:** EDA helps to improve real-time tracking and processing in logistics systems, adapting quickly to dynamic changes and improving operational efficiency.
- **IoT:** EDA in IoT systems allows for rapid adaptation to environmental changes by ensuring real-time responses, improving system efficiency and responsiveness.
- **Financial Systems:** EDA supports payment services and transaction settlement, allowing real-time processing and scalability within high-volume, high-stakes environments.

- **Old Applications (OLDI Microservices):** EDA is applied to legacy systems transitioning to microservices architectures, optimizing latency and improving scalability for modern environments.

Scenario	Industry Perspective	Academic Perspective
Financial Systems	Real-time transaction processing, fraud detection, payment processing. High volume, low margin for error.	Real-time transaction settlement, payment services, and fraud detection in financial contexts.
E-commerce	Order processing, inventory management, payment confirmation, and shipping.	Interaction between microservices for processing orders, payments, and inventory updates asynchronously.
IoT Systems	Device communication and real-time data processing for scalability and responsiveness.	Real-time responses to events with quick adaptation to environmental changes.
Social Media	Event-driven systems trigger user actions like posting, liking, and sharing to update feeds and recommend content.	Not explicitly mentioned, but relevant in the context of real-time data handling and user interaction.
Healthcare	Event streams for patient monitoring, triggering alerts and updates.	Not explicitly mentioned, but relevant for improving system responsiveness and event handling in healthcare contexts.
Supply Chain and Logistics	Inventory updates, shipment tracking, procurement flows.	Logistics optimization with event-driven systems to improve real-time tracking and responsiveness.
Real-Time Analytics	Event-based processing of real-time data for immediate insights and reporting.	Real-time analytics through event-driven interactions in distributed systems.
Ticket Management Systems	Update the state of tickets across multiple services without waiting for responses.	Not explicitly mentioned, but can be inferred under the general notion of decoupled service workflows.
Microservices and Legacy Applications	Used in large systems like microservices and SOA for scalability and fault tolerance.	Applied to optimize latency and scalability, particularly in systems transitioning to microservices.

Table 11 Scenarios Where Event-Driven Architecture (EDA) Has Been Used: Industry vs. Academia Perspectives

5 Threats to validity

In any research, it is crucial to recognize and address the potential threats to validity. In this study, we conducted a survey to gather data on specific aspects of the subject at hand. However, like any study relying on survey responses, we must acknowledge the following threats to validity:

Sampling Bias: The responses we gathered might not be fully representative of the broader population, as survey participation was limited to those who opted in. This could result in bias in our findings, especially if certain groups were underrepresented or overrepresented.

Response Bias: Participants may have provided answers based on what they thought was expected of them, or they may have misunderstood the questions. This could affect the accuracy of the data collected.

Measurement Error: The accuracy of the survey responses could be influenced by how the questions were framed, how the participants interpreted them, or external factors that could have affected their ability to provide accurate answers.

Confounding Variables: The presence of external factors that were not controlled for in the survey may have influenced the responses, impacting the internal validity of the study.

It is also important to note that all records found in the survey are available and accessible at the following link: Survey Results.

Additionally, a systematic review of the relevant literature, which was conducted as part of the research, is published in the proceedings of the ICEIS conference. You can find the full review here: Systematic Review at ICEIS Conference.

These resources provide transparency into the methodology and the results, allowing for a deeper examination of the findings and their limitations.

6 Conclusion

This study explored the application and benefits of **Event-Driven Architecture (EDA)** in various industrial and academic scenarios, demonstrating its flexibility and effectiveness in building distributed and scalable systems. Through the survey results and systematic review analysis, we observed a growing trend in the use of EDA in financial systems, e-commerce, IoT, microservices, and other high data volume systems that require low latency.

The main advantage of EDA is its ability to handle large volumes of events asynchronously, allowing systems to respond quickly and efficiently to dynamic changes. The use of EDA in e-commerce platforms, payment systems, real-time monitoring, and social networks confirms its suitability for scenarios requiring high scalability and the ability to react to events.

However, the study also revealed some limitations and threats to validity, such as sampling bias and the potential for measurement errors in the collected data. These factors may impact the generalization of the results, but they do not undermine the relevance of the conclusions. The transparency of the collected data, available through the survey link, allows for a more critical assessment of the results and a deeper discussion on the challenges faced in implementing EDA in different domains.

6.1 Future Work Opportunities

Based on the findings and limitations of this study, several areas can be explored for future work:

- **Exploration of New Use Cases and Domains**: While the study focused on e-commerce, financial systems, and IoT, other areas such as **healthcare**, **industrial automation**, and **artificial intelligence** could benefit from EDA. Future work could investigate the applicability of EDA in these contexts, focusing on how the architecture can improve communication and automation in real-time control systems.
- **Improvement in Data Validation and Quality**: An important aspect for the evolution of this work would be the **improvement in data collection quality**. Conducting a broader survey with a more diverse sample and controlling for confounding factors could contribute to greater precision and reliability in the conclusions.
- **Event Automation and Orchestration**: Event orchestration in distributed systems is a promising area. Research could focus on the development of **frameworks or tools for automating** event orchestration more effectively, especially in systems with multiple integrations.
- **Study of Resilience and Fault Tolerance**: A line of research could focus on the **resilience of EDA-based systems**. How do these systems handle failures, and how can the architecture be adjusted to ensure event processing continuity in highly distributed environments with high latency?
- **Impact on Performance and Operational Costs**: Another relevant aspect would be a deeper analysis of the **impact on performance and operational costs** when adopting EDA in large-scale systems. Investigating how event-driven architecture affects resource consumption and long-term maintenance could provide valuable insights for organizations seeking to adopt this approach.
- **EDA Integration with Other Architectures**: Future studies could investigate how **EDA can be combined with other architectures**, such as microservice-based or data-streaming systems. Evaluating the interaction between these different approaches could lead to a deeper understanding of the advantages and disadvantages of combining these architectures in a single system.
- **Development of New Consensus and Coordination Models for Events**: Research could also focus on the **development of new consensus and coordination models** in distributed systems based on EDA. This would enable greater efficiency in how events are coordinated across different components and services, improving system scalability and reliability.

In summary, the work carried out opens up several opportunities for future research, whether by improving existing practices or exploring new approaches that can benefit both the academic community and the industry at large. The evolution of EDA and its application in various scenarios will be crucial in developing more agile, scalable, and resilient systems.

7 Declarations

7.1 Funding

Not applicable.

7.2 Ethical approval

Not applicable.

7.3 Author Contributions [all authors should be mentioned]

Marcus V. S. Silva: Conceptualization, Methodology, Writing – Original Draft, Supervision.
Luiz F. C. dos Santos: Data Curation, Formal Analysis, Visualization.
Fabio Gomes Rocha: Software, Investigation, Writing – Review & Editing.
Michel S. Soares: Software, Investigation, Writing – Review.
All authors contributed equally to the development of the work. Luiz F. C. dos Santos, Fabio Gomes Rocha and Michel S. Soares contributed equally to this work. All authors have read and approved the final version of the manuscript.

7.4 Data Availability Statement

The data presented in this study are openly available at the following link: survey data

7.5 Conflict of Interest

The authors declare no conflict of interest.

7.6 Clinical Trial Number

Not applicable.

References

[1] Silva, M., dos Santos, L. F. C., Soares, M. & Rocha, F. G. *Guidelines for the application of event driven architecture in micro services with high volume of data*, 859–866. INSTICC (SciTePress, 2025).

[2] Kommera, A. R. The power of event-driven architecture: Enabling real-time systems and scalable solutions. *Turkish Journal of Computer and Mathematics Education (TURCOMAT) ISSN* **3048**, 4855 (2020).

[3] Rosa-Bilbao, J., Boubeta-Puig, J. & Rutle, A. Cepedaloco: An event-driven architecture for integrating complex event processing and blockchain through low-code. *Internet of Things* **22**, 100802 (2023).

[4] Qiao, X. *et al.* An edsoa-based services provisioning approach for internet of things. *Sci. China Inf. Sci* **43**, 1219–1243 (2013).

[5] Aman, W. & Snekkenes, E. Edas: An evaluation prototype for autonomic event-driven adaptive security in the internet of things. *Future Internet* **7**, 225–256 (2015).

[6] Rahmani, A. M., Babaei, Z. & Souri, A. Event-driven iot architecture for data analysis of reliable healthcare application using complex event processing. *Cluster Computing* **24**, 1347–1360 (2021).

[7] Keele, S. *et al.* Guidelines for performing systematic literature reviews in software engineering. Tech. Rep., Technical report, ver. 2.3 ebse technical report. ebse (2007).

[8] Petersen, K., Feldt, R., Mujtaba, S. & Mattsson, M. *Systematic mapping studies in software engineering* (BCS Learning & Development, 2008).

[9] Parsifal. Systematic literature review tool. https://parsif.al (2024). Available at: https://parsif.al.

[10] Pan, G.-C., Liu, P. & Wu, J.-J. *A cloud-native online judge system*, 1293–1298 (IEEE, 2022).

[11] Kitchenham, B. & Brereton, P. A systematic review of systematic review process research in software engineering. *Information and software technology* **55**, 2049–2075 (2013).

[12] Laigner, R., Zhou, Y. & Salles, M. A. V. *A distributed database system for event-based microservices*, 25–30 (2021).

[13] Pour, S., Balouek, D., Masoumi, A. & Brynskov, M. *An urgent computing oriented architecture for dynamic climate risk management framework*, 1–4 (2023).

[14] Ren, Z. *et al. Migrating web applications from monolithic structure to microservices architecture*, 1–10 (2018).

[15] Dinh-Tuan, H., Mora-Martinez, M., Beierle, F. & Garzon, S. R. *Development frameworks for microservice-based applications: Evaluation and comparison*, 12–20 (2020).

[16] Henning, S. & Hasselbring, W. Theodolite: Scalability benchmarking of distributed stream processing engines in microservice architectures. *Big Data Research* **25**, 100209 (2021).

[17] Laigner, R. *et al. From a monolithic big data system to a microservices event-driven architecture*, 213–220 (IEEE, 2020).

[18] Zhelev, S. & Rozeva, A. *Using microservices and event driven architecture for big data stream processing*, Vol. 2172 (AIP Publishing, 2019).

[19] Raj, P., Vanga, S. & Chaudhary, A. *Cloud-Native Computing: How to Design, Develop, and Secure Microservices and Event-Driven Applications* (John Wiley & Sons, 2022).

[20] García, M. M. & Anglin. *Learn Microservices with Spring Boot* (Springer, 2020).

[21] Rahmatulloh, A., Nugraha, F., Gunawan, R. & Darmawan, I. *Event-driven architecture to improve performance and scalability in microservices-based systems*, 01–06 (IEEE, 2022).

[22] Zuki, S. Z. M., Mohamad, R. & Saadon, N. A. Containerized event-driven microservice architecture. *Baghdad Science Journal* **21**, 0584–0584 (2024).

[23] Laigner, R., Zhou, Y., Salles, M. A. V., Liu, Y. & Kalinowski, M. Data management in microservices: State of the practice, challenges, and research directions. *arXiv preprint arXiv:2103.00170* (2021).

[24] Figueira, J. & Coutinho, C. Developing self-adaptive microservices. *Procedia Computer Science* **232**, 264–273 (2024).

[25] Aksakalli, I. K., Çelik, T., Can, A. B. & Tekinerdoğan, B. Deployment and communication patterns in microservice architectures: A systematic literature review. *Journal of Systems and Software* **180**, 111014 (2021).

[26] Wöhrer, M., Zdun, U. & Rinderle-Ma, S. *Architecture design of blockchain-based applications*, 173–180 (IEEE, 2021).

[27] Singjai, A., Zdun, U., Zimmermann, O. & Pautasso, C. *Patterns on deriving apis and their endpoints from domain models*, 1–15 (2021).

[28] Mathew, A., Andrikopoulos, V., Blaauw, F. J. & Karastoyanova, D. Pattern-based serverless data processing pipelines for function-as-a-service orchestration systems. *Future Generation Computer Systems* **154**, 87–100 (2024).

[29] Alulema, D., Criado, J., Iribarne, L., Fernández-García, A. J. & Ayala, R. Si4iot: A methodology based on models and services for the integration of iot systems. *Future Generation Computer Systems* **143**, 132–151 (2023).

[30] Medeiros, A. Dynamics of change: Why reactivity matters: Tame the dynamics of change by centralizing each concern in its own module. *Queue* **14**, 68–82 (2016).

[31] Vangala, S. R., Kasimani, B. & Mallidi, R. K. *Microservices event driven and streaming architectural approach for payments and trade settlement services*, 1–6 (IEEE, 2022).

[32] *SCOPES '21: Proceedings of the 24th International Workshop on Software and Compilers for Embedded Systems* (Association for Computing Machinery, New York, NY, USA, 2021).

[33] Tovarnitchi, V. M. *Designing distributed, scalable and extensible system using reactive architectures*, 484–488 (IEEE, 2019).

[34] Hustad, E. & Olsen, D. H. Creating a sustainable digital infrastructure: The role of service-oriented architecture. *Procedia Computer Science* **181**, 597–604 (2021).

[35] Monteiro, D., Gadelha, R., Maia, P. H. M., Rocha, L. S. & Mendonça, N. C. *Beethoven: an event-driven lightweight platform for microservice orchestration*, 191–199 (Springer, 2018).

[36] Kuhn, M. & Franke, J. *Smart manufacturing traceability for automotive e/e systems enabled by event-driven microservice architecture*, 142–148 (IEEE, 2020).

[37] Liu, X. & Buyya, R. Resource management and scheduling in distributed stream processing systems: a taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)* **53**, 1–41 (2020).

[38] Wu, C.-F., Ma, S.-P., Shau, A.-C. & Yeh, H.-W. *Testing for event-driven microservices based on consumer-driven contracts and state models*, 467–471 (IEEE, 2022).

[39] Khalloof, H. *et al.* *A generic distributed microservices and container based framework for metaheuristic optimization*, 1363–1370 (2018).

[40] Sicari, C., Balouek, D., Parashar, M. & Villari, M. *Event-driven faas workflows for enabling iot data processing at the cloud edge continuum*, 1–10 (2023).

[41] Romanov, O., Mankivskyi, V., Saychenko, I. & Nesterenko, M. *Event model algorithm with microservice architecture implementation*, 561–566 (IEEE, 2022).

[42] Singh, A., Singh, V., Aggarwal, A. & Aggarwal, S. *Event driven architecture for message streaming data driven microservices systems residing in distributed version control system*, 308–312 (IEEE, 2022).

[43] Xie, R., Yang, J., Li, J. & Wang, L. *Impacttracer: root cause localization in microservices based on fault propagation modeling*, 1–6 (IEEE, 2023).

[44] Lin, C., Khazaei, H., Walenstein, A. & Malton, A. Autonomic security management for iot smart spaces. *ACM Transactions on Internet of Things* **2**, 1–20 (2021).

[45] Chenouf, M. A. & Aissaoui, M. *An event-driven architecture (eda) adapted to cloud-based hospital information systems (his)*, 651–659 (Springer, 2022).

[46] Kniazhyk, T. & Muliarevych, O. Cloud computing with resource allocation based on ant colony optimization. *Advances in Cyber-Physical Systems* **8**, 104 (2022).

[47] Leveling, J., Weickhmann, L., Nissen, C. & Kirsch, C. *Event-driven architecture for sensor data integration for logistics services*, 536–540 (IEEE, 2018).