Guidelines for the Application of Event Driven Architecture in Micro Services with High Volume of Data

Marcus V. S. Silva[a], Luiz F. C. dos Santos[b], Michel S. Soares[c] and Fabio Gomes Rocha[d]
Federal University of Sergipe, Av. Mal. Cândido Rondon, Rosa Elze, 1861 - São Cristóvão, Sergipe, Brazil

Keywords: Event-Driven Architecture (EDA), Microservices, Scalability, Responsiveness, Decoupling, Real-Time Processing, State Management, Software Architecture.

Abstract: Event-Driven architecture (EDA) has proven itself as a transformative strategy within microservices, celebrated for its role in enabling scalable, responsive, and decoupled interactions among system components. This paper draws on insights from diverse domains such as e-commerce, healthcare, IoT, and data processing to showcase how EDA can revolutionize system agility and responsiveness, particularly under high communication loads and real-time processing demands. We underscore EDA's effectiveness in optimizing critical processes like order management, payment processing, and real-time anomaly detection through 13 case studies. These enhancements not only boost operational efficiency but also foster more informed decision-making. Moreover, the burgeoning interest in applying EDA to complex systems that necessitate dynamic adaptation to environmental changes, such as climate risk management and intelligent manufacturing, underscores its potential. However, adopting EDA is not without its challenges, particularly in state management, consistency, and testing, which necessitate further exploration. This paper contributes to the discourse on EDA by reviewing its current state, challenges, and future directions, offering a comprehensive perspective on its role and potential in modern software architecture.

1 INTRODUCTION

EDA has emerged as an essential approach in developing distributed systems, especially in contexts where agility and flexibility are crucial for organizational competitiveness. As application complexity grows and the demand for scalable solutions increases, EDA provides a resilient framework for building systems responsive to dynamic events. EDA enables different system components to communicate asynchronously, fostering a smoother and more reactive interaction between microservices.

EDA has proven to be indispensable in the e-commerce sector (Gördesli and Varol, 2022) due to the need for continuous communication among microservices, including order processing, payments, and inventory control, which rely on dynamic event exchanges. EDA significantly enhances operational efficiency and enables more effective information flow management, a critical factor in maintaining competitiveness in a constantly evolving market. Considering the transition from monolithic applications to microservice architectures, EDA has been widely adopted, as it simplifies migration and provides an organized, scalable framework for intensive microservice communication.

2 EVENT-DRIVEN ARCHITECTURE

EDA has emerged as one of the leading approaches for developing distributed systems, especially those that require high scalability, flexibility, and real-time responsiveness (Figueira and Coutinho, 2024). EDA is based on asynchronous communication between system components through the production, detection, and consumption of events, which allows for constructing more decoupled and reactive systems (Laigner et al., 2021b). Monolithic architectures face challenges related to scalability and maintenance difficulty because their components are tightly coupled

[a] https://orcid.org/0009-0000-9211-5259
[b] https://orcid.org/0000-0003-4538-5410
[c] https://orcid.org/0000-0002-7193-5087
[d] https://orcid.org/0000-0002-0512-5406

Silva, M. V. S., Santos, L. F. C., Soares, M. S. and Rocha, F. G.
Guidelines for the Application of Event Driven Architecture in Micro Services with High Volume of Data.
DOI: 10.5220/0013348600003929

In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 2*, pages 859-866
ISBN: 978-989-758-749-8; ISSN: 2184-4992

(Laigner et al., 2020), and any change in one part of the system may require modifications in other areas. EDA offers a solution to these problems, allowing components to be decoupled and scaled independently.

EDA's principal characteristic is that systems based on this model can react to real-time events, which is particularly relevant in scenarios such as large-scale data processing, the Internet of Things (IoT), and financial applications. For instance, in a payment system, the completion of a transaction can be treated as an event, triggering different services without waiting for synchronous responses. This conduct reduces latency and improves user experience in applications requiring fast responses.

However, despite its many benefits, EDA presents challenges that must be carefully managed, such as data consistency in distributed systems and the complexity of state management. The asynchronous nature of EDA can make it challenging to ensure that all events are processed correctly and that systems maintain consistency during failures. Monitoring and tracking real-time events also adds complexity to implementation and maintenance.

3 RELATED WORK

The cost analysis associated with EDA is also an important focus, as discussed by (Cabane and Farias, 2024), who conducted a comparative study to understand the financial impacts of this approach in different scenarios (Cabane and Farias, 2024). Furthermore, data management in microservices, including challenges and research directions, is explored by (Laigner et al., 2021b), providing valuable insights into how to deal with the complexity of data in event-driven architectures.

(Kniazhyk and Muliarevych, 2022) discuss the benefits and challenges of EDA in the context of cloud computing, highlighting future research directions in this area and suggesting the need for a deeper understanding of practical implementations (Raj et al., 2022). Additionally, the integration of static and dynamic data in a hybrid traceability model, as presented by (Kuhn et al., 2022), is an example of how EDA can be applied to improve transparency and traceability in complex processes.

The development of adaptive microservices, discussed by (Figueira and Coutinho, 2024), emphasizes the importance of adaptability in event-driven systems, aligning with the need for flexibility in production environments (Giovanni and Manuaba, 2022). On the other hand, (Henning and Hasselbring, 2021) analyze performance of distributed stream processing mechanisms, presenting benchmarks that can guide developers in choosing the right tools for their applications.

4 METHODOLOGY

This article aims to characterize, in a structured way, from a broad perspective, a guideline for EDA. 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 (Keele et al., 2007). The systematic mapping presented in this article follows the guidelines of Kitchenham and Charters (Petersen et al., 2008) 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 (Parsifal, 2024).

A detailed protocol was followed to classify the articles, ensuring the study's transparency and reproducibility. 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.

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''),`

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

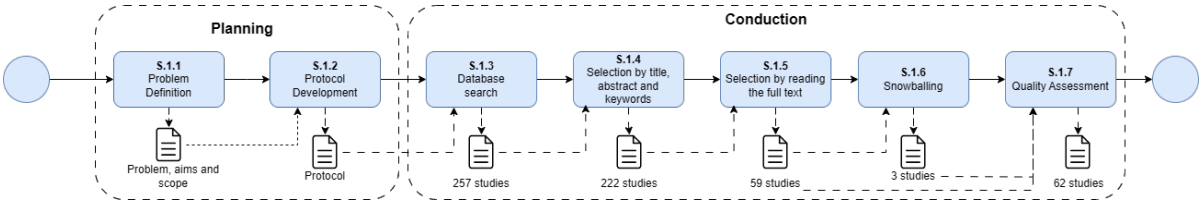

Figure 1: Selection of Articles.

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 in (Pan et al., 2022). 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 (Kitchenham and Brereton, 2013), 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.

5 RESULTS

Each research question was thoroughly examined to extract significant insights into the benefits, challenges, and trends of this architecture. Findings were divided into subthemes to facilitate understanding and visualization of the information. In each subtheme, answers to the specific questions are presented.

5.1 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 (Laigner et al., 2021a) and facilitating the efficient processing of large volumes of data (Pour et al., 2023). Modularity and decoupling enable new services to be integrated into the system without significantly impacting the existing environment (Laigner et al., 2021a).

The flexibility of EDA is reflected in the possibility of using different technology stacks for each service (Ren et al., 2018), as well as the independence of updates, allowing each service to be updated independently without affecting the entire system (Dinh-Tuan et al., 2020).

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.

5.2 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 (Laigner et al., 2021a). Similarly, systems facing performance bottlenecks in specific modules can benefit from migrating to an event-driven architecture (Ren et al., 2018).

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%

EDA also excels in solutions that require distributed processing and interactions among multiple components, ensuring that communication between them is efficient and scalable (Khalloof et al., 2018). 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 (Sicari et al., 2023).

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 (Zhelev and Rozeva, 2019).

5.3 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, (Ren et al., 2018) points out that, in small systems, using EDA can generate unnecessary overhead, particularly regarding communication Between services and network latency, (Laigner et al., 2021a) 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 (Romanov et al., 2022), 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 (Henning and Hasselbring, 2021), applying EDA in scenarios where events are not central to the system requirements can lead to scalability difficulties and increased operational costs.

5.4 RQ4. Do Engineers Have any Guidelines for Selecting Event-Driven Architecture Today?

According to (Ren et al., 2018), engineers should consider systems that require scalability, component autonomy, and real-time response as potential candidates for implementing EDA. Additionally, (Khalloof et al., 2018) highlights the need for scalability, parallel execution, and asynchronous communication as determining factors for using EDA.

On the other hand, (Pour et al., 2023) mentions that EDA is particularly useful in systems dealing with large volumes of real-time data, such as in extreme weather events. Additionally, (Singh et al., 2022) reinforces the use of EDA in scenarios involving high scalability, real-time event processing, and service decoupling.

Moreover,(Tovarnitchi, 2019) 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.

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.	(Henning and Hasselbring, 2021), (Laigner et al., 2020), (Zhelev and Rozeva, 2019), (Raj et al., 2022), (García and Anglin, 2020), (Rahmatulloh et al., 2022), (Zuki et al., 2024).
Decoupling	Facilitates decoupling between services, promoting a modular architecture that simplifies system maintenance and evolution.	(Laigner et al., 2021b), (Figueira and Coutinho, 2024), (Aksakalli et al., 2021), (Laigner et al., 2020), (García and Anglin, 2020), (Wöhrer et al., 2021)
Flexibility	Supports multiple technology stacks in different services, allowing updates and changes without impacting the entire system.	(Figueira and Coutinho, 2024), (Pan et al., 2022), (Singjai et al., 2021), (Mathew et al., 2024), (Alulema et al., 2023), (Laigner et al., 2020), (García and Anglin, 2020)
Reactivity	Facilitates real-time response to events, improving the system's ability to react quickly to environmental changes.	(Medeiros, 2016), (Vangala et al., 2022), (10., 2021), (Tovarnitchi, 2019)
Resilience	The architecture continues operating even with individual microservices failures, increasing fault tolerance and system robustness.	(Hustad and Olsen, 2021), (Monteiro et al., 2018), (Kuhn and Franke, 2020), (Liu and Buyya, 2020), (Wu et al., 2022)

5.5 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 (Xie et al., 2023). The need to ensure that event-based constraints are correctly applied is one of the points highlighted by (Laigner et al., 2021a). The difficulty of dealing with latency and the communication overhead between services is also a recurring problem, as described by (Ren et al., 2018).

Another challenge is state management, which requires ensuring data consistency across multiple services, particularly in distributed scenarios (Laigner et al., 2021b). Additionally, event logging and tracking across different services can be complicated due to the asynchronous nature of EDA, as indicated by (Lin et al., 2021).

Finally, the challenge of ensuring data security in transit, especially in distributed environments, is emphasized in (Chenouf and Aissaoui, 2022).

5.6 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 (Kniazhyk and Muliarevych, 2022). 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 (Alulema et al., 2023).

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.

5.7 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 au-

thentication 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 (Singh et al., 2022).

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 (Laigner et al., 2021a). 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 (Romanov et al., 2022).

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.

5.8 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 (Sicari et al., 2023) or logistics services how demonstrated (Leveling et al., 2018). In IoT systems, EDA enables real-time responses to events by promoting an architecture that adapts to rapid changes in the environment (Lin et al., 2021).

In the financial context, EDA is applied in payment services and transaction settlement, demonstrating its effectiveness in real-time information processing (Vangala et al., 2022).

6 DISCUSSION

EDA has proven to be a practical and flexible approach to building modern systems, especially in environments where scalability and resilience are crucial. The discussed benefits, such as scalability, decoupling, flexibility, low latency, and reactivity, demonstrate EDA's ability to meet the dynamic demands of contemporary applications. Additionally, the architecture's resilience is a crucial factor, as it allows the system to continue operating even in the face of failures in individual microservices, enhancing the system's robustness and reliability.

However, it is essential to highlight the need for a deeper exploration of two critical aspects: costs and security. Analyzing the costs associated with implementing and maintaining EDA is fundamental for organizations seeking to adopt this architecture. The initial setup and training costs can often be high, especially if the team is unfamiliar with EDA concepts. Furthermore, operating multiple microservices can result in additional expenses, such as communication costs between services and infrastructure management. Thus, a thorough evaluation of costs should be a priority to ensure that the expected benefits outweigh the incurred expenses.

Security also represents a significant challenge in adopting EDA. The distributed nature of microservices increases the attack surface, making the system more vulnerable to threats and cyberattacks, as explored by (Molina et al., 2018).

To illustrate all the points discussed, Table 5 has been created, summarizing the main benefits, challenges, and considerations related to EDA. This visual resource is a valuable tool to help stakeholders better understand the complexity and nuances of this architecture.

Table 5: Recommended Style Guide for EDA.

Aspect	Technologies and Considerations
Database	NoSQL: MongoDB; SQL: SQL Server, Oracle, MySQL.
Application Domains	IoT, Flight Simulation, Book Retail, Big Data, E-commerce, Financial Systems.
Key Benefits	Availability, Scalability, Elasticity, Small Service Modularity.
Challenges	Traceability, Observability, Rollback Process, Data Consistency (Strong and Eventual), Saga Patterns (Choreography and Orchestration), Security.
Patterns	CQRS, Event Sourcing.
Message Brokers	Kafka, RabbitMQ, Service Bus.

7 CONCLUSIONS

EDA significantly evolves how software systems are designed and implemented. As discussed, EDA enhances scalability and flexibility and promotes excel-

lent decoupling between services, which is essential for building modern and resilient applications. EDA is particularly relevant in high-load scenarios and systems that require real-time responses, as evidenced by research on microservices and distributed systems.

However, the adoption of EDA should be accompanied by a careful analysis of costs and security challenges. The costs associated with implementing EDA can be high, especially in organizations needing more microservices experience. Companies must conduct a cost-benefit analysis before migrating to this architecture, considering the initial costs and ongoing maintenance and operation expenses.

Furthermore, security is a critical aspect that requires special attention. The distributed nature of EDA can increase the vulnerability of systems to cyberattacks. Thus, organizations must establish robust security guidelines covering authentication, authorization, and encryption to protect data and ensure user privacy.

In summary, EDA is not just a technical solution but a strategic approach that, when effectively implemented, can provide significant competitive advantages. However, organizations need to continue investigating EDA's financial and security implications to maximize benefits while minimizing associated risks. Future research should focus on developing frameworks and guidelines that help organizations navigate cost and security issues, ensuring that EDA is a viable and secure choice for software development.

REFERENCES

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

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

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

Cabane, H. and Farias, K. (2024). On the impact of event-driven architecture on performance: An exploratory study. *Future Generation Computer Systems*, 153:52–69.

Chenouf, M. A. and Aissaoui, M. (2022). An event-driven architecture (eda) adapted to cloud-based hospital information systems (his). In *Proceedings of Sixth International Congress on Information and Communication Technology: ICICT 2021, London, Volume 1*, pages 651–659. Springer.

Dinh-Tuan, H., Mora-Martinez, M., Beierle, F., and Garzon, S. R. (2020). Development frameworks for microservice-based applications: Evaluation and comparison. In *Proceedings of the 2020 European Symposium on Software Engineering*, pages 12–20.

Figueira, J. and Coutinho, C. (2024). Developing self-adaptive microservices. *Procedia Computer Science*, 232:264–273.

García, M. M. and Anglin (2020). *Learn Microservices with Spring Boot*. Springer.

Giovanni, E. D. and Manuaba, I. B. K. (2022). Event-driven approach in microservices architecture for flight booking simulation. *ICIC Express Letters*, 16(5):545–553.

Gördesli, M. and Varol, A. (2022). Comparing interservice communications of microservices for e-commerce industry. In *2022 10th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–4. IEEE.

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

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

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

Khalloof, H., Jakob, W., Liu, J., Braun, E., Shahoud, S., Duepmeier, C., and Hagenmeyer, V. (2018). A generic distributed microservices and container based framework for metaheuristic optimization. In *Proceedings of the genetic and evolutionary computation conference companion*, pages 1363–1370.

Kitchenham, B. and Brereton, P. (2013). A systematic review of systematic review process research in software engineering. *Information and software technology*, 55(12):2049–2075.

Kniazhyk, T. and Muliarevych, O. (2022). Cloud computing with resource allocation based on ant colony optimization. *Advances in Cyber-Physical Systems*, 8(2):104.

Kuhn, M. and Franke, J. (2020). Smart manufacturing traceability for automotive e/e systems enabled by event-driven microservice architecture. In *2020 IEEE 11th International Conference on Mechanical and Intelligent Manufacturing Technologies (ICMIMT)*, pages 142–148. IEEE.

Kuhn, M., Kaminski, E. T., and Franke, J. (2022). Track and trace: Integrating static and dynamic data in a hybrid graph-based traceability model. *Procedia CIRP*, 112:250–255.

Laigner, R., Kalinowski, M., Diniz, P., Barros, L., Cassino, C., Lemos, M., Arruda, D., Lifschitz, S., and Zhou, Y. (2020). From a monolithic big data system to a microservices event-driven architecture. In *2020 46th Euromicro conference on software engineering and advanced applications (SEAA)*, pages 213–220. IEEE.

Laigner, R., Zhou, Y., and Salles, M. A. V. (2021a). A distributed database system for event-based microservices. In *Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems*, pages 25–30.

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

Leveling, J., Weickhmann, L., Nissen, C., and Kirsch, C. (2018). Event-driven architecture for sensor data integration for logistics services. In *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 536–540. IEEE.

Lin, C., Khazaei, H., Walenstein, A., and Malton, A. (2021). Autonomic security management for iot smart spaces. *ACM Transactions on Internet of Things*, 2(4):1–20.

Liu, X. and Buyya, R. (2020). Resource management and scheduling in distributed stream processing systems: a taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 53(3):1–41.

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

Medeiros, A. (2016). Dynamics of change: Why reactivity matters: Tame the dynamics of change by centralizing each concern in its own module. *Queue*, 14(3):68–82.

Molina, J. M., García, J. F., and Jiménez, C. K. (2018). Archer: An event-driven architecture for cyber-physical systems. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 335–340. IEEE.

Monteiro, D., Gadelha, R., Maia, P. H. M., Rocha, L. S., and Mendonça, N. C. (2018). Beethoven: an event-driven lightweight platform for microservice orchestration. In *Software Architecture: 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24–28, 2018, Proceedings 12*, pages 191–199. Springer.

Pan, G.-C., Liu, P., and Wu, J.-J. (2022). A cloud-native online judge system. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1293–1298. IEEE.

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

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *12th international conference on evaluation and assessment in software engineering (EASE)*. BCS Learning & Development.

Pour, S., Balouek, D., Masoumi, A., and Brynskov, M. (2023). An urgent computing oriented architecture for dynamic climate risk management framework. In *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, pages 1–4.

Rahmatulloh, A., Nugraha, F., Gunawan, R., and Darmawan, I. (2022). Event-driven architecture to improve performance and scalability in microservices-based systems. In *2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)*, pages 01–06. IEEE.

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

Ren, Z., Wang, W., Wu, G., Gao, C., Chen, W., Wei, J., and Huang, T. (2018). Migrating web applications from monolithic structure to microservices architecture. In *Proceedings of the 10th Asia-Pacific Symposium on Internetware*, pages 1–10.

Romanov, O., Mankivskyi, V., Saychenko, I., and Nesterenko, M. (2022). Event model algorithm with microservice architecture implementation. In *2022 IEEE 9th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)*, pages 561–566. IEEE.

Sicari, C., Balouek, D., Parashar, M., and Villari, M. (2023). Event-driven faas workflows for enabling iot data processing at the cloud edge continuum. In *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, pages 1–10.

Singh, A., Singh, V., Aggarwal, A., and Aggarwal, S. (2022). Event driven architecture for message streaming data driven microservices systems residing in distributed version control system. In *2022 International Conference on Innovations in Science and Technology for Sustainable Development (ICISTSD)*, pages 308–312. IEEE.

Singjai, A., Zdun, U., Zimmermann, O., and Pautasso, C. (2021). Patterns on deriving apis and their endpoints from domain models. In *Proceedings of the 26th European Conference on Pattern Languages of Programs*, pages 1–15.

Tovarnitchi, V. M. (2019). Designing distributed, scalable and extensible system using reactive architectures. In *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pages 484–488. IEEE.

Vangala, S. R., Kasimani, B., and Mallidi, R. K. (2022). Microservices event driven and streaming architectural approach for payments and trade settlement services. In *2022 2nd International Conference on Intelligent Technologies (CONIT)*, pages 1–6. IEEE.

Wöhrer, M., Zdun, U., and Rinderle-Ma, S. (2021). Architecture design of blockchain-based applications. In *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 173–180. IEEE.

Wu, C.-F., Ma, S.-P., Shau, A.-C., and Yeh, H.-W. (2022). Testing for event-driven microservices based on consumer-driven contracts and state models. In *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, pages 467–471. IEEE.

Xie, R., Yang, J., Li, J., and Wang, L. (2023). Impacttracer: root cause localization in microservices based on fault propagation modeling. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE.

Zhelev, S. and Rozeva, A. (2019). Using microservices and event driven architecture for big data stream processing. In *AIP Conference Proceedings*, volume 2172. AIP Publishing.

Zuki, S. Z. M., Mohamad, R., and Saadon, N. A. (2024). Containerized event-driven microservice architecture. *Baghdad Science Journal*, 21(2 (SI)):0584–0584.