Event-driven Architecture

Bachelorseminararbeit

Bachelorseminar Wintersemester 2022

Name: Noel Röhrig (7380550)

Betreuer: Pascal Budner, Felix Döppers

Studiengang: Wirtschaftsinformatik

Seminar für Wirtschaftsinformatik und Informationsmanagement

Wirtschafts- und Sozialwissenschaftliche Fakultät

Universität zu Köln

Köln, 28.10.2022

Table of Contents

Table of Figures

Table of Tables

Table of Abbreviations

[EDA]	[Event-driven Architecture]
[IoT]	[Internet of Things]
[IIoT]	[Industrial Internet of Things]
[SOA]	[Service-oriented Architecture]
[CEP]	[Complex Event Processing]
[PtP]	[Peer-to-Peer]

1 Introduction

The amount of data that is generated by devices such as sensors or mobile phones keeps increasing heavily and is projected to reach 79,4 zettabytes by 2025 each day (Laigner et al., 2020, Page 1). Managing and storing data from Internet of Things (IoT) sensors to discover the value of said data is an important issue (Chen et al., 2014, Page 1).

“The challenge to manage data, transform it into knowledge and make smart automated decisions, has drawn a lot of attention” (Theorin et al., 2017, Page 1).

This creates a need to find adequate principles and patterns to create high responsive, oftentimes nearly real-time, systems (Venkatesan and Sridhar, 2017, Page 490). Event-Driven Architecture (EDA), with its extreme loose coupling and asynchronous approach helps businesses to overcome these challenges with various styles of processing events (Michelson, 2006, Pages 3-4). While being a de facto industry standard approach to new software systems, there are scientific works on various specifications of EDA application fields such as Internet of Things (IoT), manufacturing, trade systems or industry 4.0 and industrial Internet of things (IIoT) (Chen et al., 2014; Laliwala and Chaudhary, 2008; Theorin et al., 2017; Zhang et al., 2014). But the current literature lacks a clear overview on EDA in general and various approaches and their benefits and challenges.

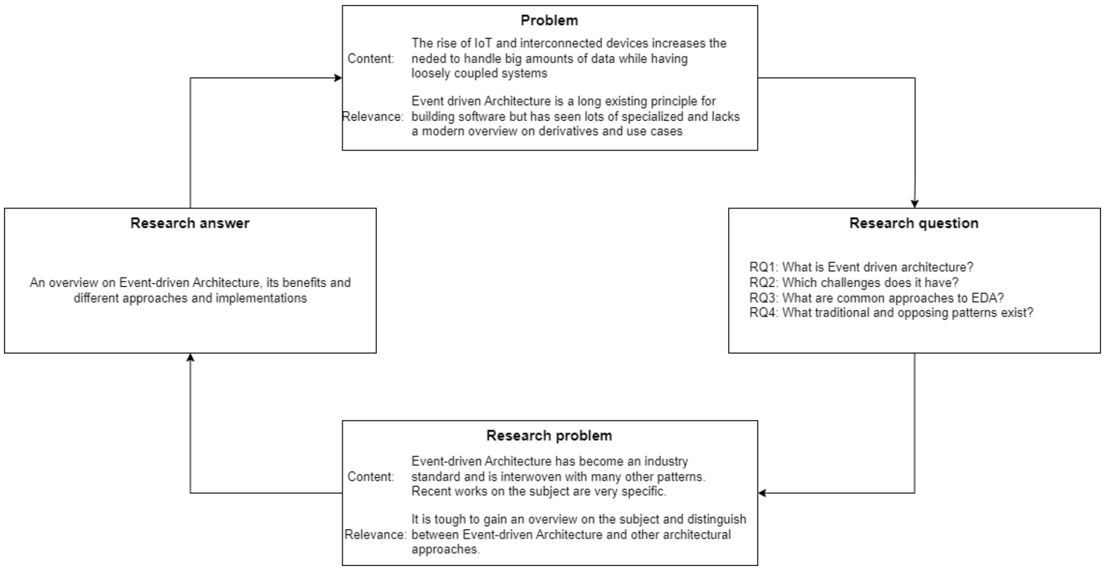
Problem
Content: The rise of IoT and interconnected devices increases the neded to handle big amounts of data while having loosely coupled systems
Relevance: Event driven Architecture is a long existing principle for building software but has seen lots of specialized and lacks a modern overview on derivatives and use cases
Research question
RQ1: What is Event driven architecture?
RQ2: Which challenges does it have?
RQ3: What are common approaches to EDA?
RQ4: What traditional and opposing patterns exist?
Research problem
Content: Event-driven Architecture has become an industry standard and is interwoven with many other patterns. Recent works on the subject are very specific.
Relevance: It is tough to gain an overview on the subject and distinguish between Event-driven Architecture and other architectural approaches.
Research answer
An overview on Event-driven Architecture, its benefits and different approaches and implementations

Figure 1: Research cycle

The intention with this work is to clarify the current industry situation on EDA and its different approaches by giving an overview on EDA in general, different processing

styles (Michelson, 2006, Pages 3-4) and deep diving into differences of suggested solutions for specific problem domains. Since EDA implementations often correlate with other architectural patterns and principles (Laigner et al., 2020; Michelson, 2006; Theorin et al., 2017; Zhang et al., 2014), most prominently Service-oriented Architecture (SOA) and microservices, a short introduction into those and their benefits will be necessary.
This work can therefore be beneficial in creating an outline on the subject. It also helps in finding suitable approaches to EDA for different problem domains. The deployed research questions are as follows:

- **RQ1: What is Event-driven Architecture?**
- **RQ2: What are common challenges and benefits of EDA?**
- **RQ3: What are common approaches to EDA?**
- **RQ4: What traditional and opposing patterns exist?**

The resulting paper will consist of the following: After this introduction, section 2 defines key terminology and the basics of EDA and some of its application fields. Section 3 then provides an insight on the methodology of this literature review. Furthermore, section 4 will contain the results of the methodology and provide a concept matrix to better classify the underlying literature. Finally, section V will present the implications and results of this work and give an outline for future works on the subject before section VI concludes this paper.

2 Theoretical background

To provide a better understanding of the discussed terms and concepts, a basic introduction to various topics is given below.

2.1 Event-driven Architecture

EDA is an asynchronous and data-centric approach to communication within a system (Levina and Stantchev, 2009, Page 37). The systems aim to raise an event whenever a notable thing happens. Downstream, interested parties then evaluate the given events, and may trigger a business process or direct treatment through a software system (Michelson, 2006, Page 3). This decouples the primary, event raising system from any references or knowledge to its dependent systems (Levina and Stantchev, 2009, Page 38). The event recipients (or consumers) can then process these events in a non-linear and dynamic manner (State Grid Electric Power Research Institute et al., 2018, Page 254) and perform their functionality or publish an alert (Levina and Stantchev, 2009, Page 38).

2.1.1 Event

An event contains two parts: the meta data about the event (or event header) and an event body. The event header describes the event occurrence and contains information such as the originator, the name, and an ID (among others). On the other hand, the event body contains all relevant information of the event itself (Michelson, 2006, Page 3). Events can be concrete with a direct manifestation in reality (such as a sensor alert) or inferred which can only be inferred from the context and a history of concrete events (Magid et al., 2008, Page 253).
Furthermore the term event can be used interchangeably to describe the specification (or definition) as well as the individual occurrence (or instance) of the event (Michelson, 2006, Page 3)

2.1.2 Simple Event & Stream Event processing

Event processing is a way of keeping track of events that occur in a physical system and analyzing this data (State Grid Electric Power Research Institute et al., 2018, Page 251). If these events restrict to being notable events such as low inventory alerts, the system is categorized as Simple event processing. If the raised events also include ordinary events without a direct value or use, the system is categorized as Stream Event Processing (Michelson, 2006, Pages 3-4).

2.1.3 Complex event processing

Adding onto stream event processing, complex events can be described as a group of events that “contain elements from different contexts or different time points” (Levina and Stantchev, 2009, Page 38) but still correlate. Furthermore complex event processing (CEP) describes the technique to evaluate this confluence of events (Michelson, 2006, Page 4). The complexity comes from identifying meaningful events, recognize business interests and respond in real-time (State Grid Electric Power Research Institute et al., 2018, Page 251).

2.2 Software Patterns

Since EDA correlates well with certain architectural patterns and displace traditional approaches, they need to be briefly explained.

2.2.1 SOA

SOA is an approach to decomposing a software system in its functionalities to create an autonomous and platform-independent entity (Levina and Stantchev, 2009, Page 38). Therefore, SOA deconstructs a system by decomposing it into loosely coupled services (Lan et al., 2015, Page 5) resulting in “ubiquity and functional independence” (Levina and Stantchev, 2009, Page 37). Applying SOA means extracting and encapsulating business processes so they are reusable through the resulting service (Levina and Stantchev, 2009, Pages 37-38) which is a self-contained application (Theorin et al., 2017, Page 1298).

2.2.2 Client-Server pattern

Connecting systems is traditionally done using a Peer-to-Peer (PtP) and client/server approach. This means explicitly creating a connection between the server and client and having to update all dependent applications to integrate new applications. In a fully connected network this results in quadratic number of connections in relation to applications (Theorin et al., 2017, Page 1298).

2.2.3 Publish-Subscribe Pattern

The Publish-Subscribe pattern consists of a subscriber and a publisher. The subscriber connects to a feed provided by a middleware or the publisher directly. The publisher feeds information either to the middleware or directly to the subscriber. Through this approach, message delivery and consumption are decoupled, and an asynchronous delivery of messages is ensured. Furthermore, the provided information can be consumed by a multitude of subscribers. This approach is often the basis for EDA (Rieke et al., 2018, Pages 3-4).

2.2.4 Monolith

A monolith is the result of traditional software development, in which modules, subsystems and functionalities are integrated in a centralized manner. In a monolith, several aspects of a software system are bundled in a single application, creating a large source code. The size of a monolith, high degree of coupling and the natural corrosion of software quality create a limitation on scalability and maintainability.

Furthermore, the adoption of new technologies is undermined by a monolithic structure (Laigner et al., 2020, Pages 1-2).

2.3 Application Fields

By providing a sustainable evolution of underlying software systems with its loose coupling and open-ended extension approach (Chiprianov et al., 2014, Page 360), EDA has a broad application field since it thus enables manageable growth.

2.3.1 IoT

IoT creates a connection between the real world and the digital world by introduction of mass sensors (Lan et al., 2015, Page 4). These sensors generate unprecedented volumes of heterogenous, near real-time data which need to be utilized and processed (Rieke et al., 2018, Page 2).

2.3.2 Industry 4.0

Propelled by IoT and Services, manufacturing experiences its fourth industrial revolution. Businesses create a global network of their machinery, warehouse systems and production facilities ("Industrie 4.0 Working Group," 2013, Page 5). But this generates the challenge to manage and process the new data to make smart automated decisions and create knowledge (Theorin et al., 2017, Page 1297).

3 Methodology

This literature review adheres to the criteria of Webster and Watson (Webster and Watson, 2002) and therefore started with a broad search in multiple journals and databases. Furthermore, the results are classified by key concepts and organized in a concept matrix, enabling a better analysis of the literature.

3.1 Finding literature

The aim of this literature review is to create an overview and insight on Event-driven Architecture. Therefore, to get a general idea of the topic and the state of its literature, the initial search string used in Google Scholar and ProQuest was just "Event-driven Architecture". Two key results emerged from scanning the first few results. Firstly, Event-driven Architecture as an approach to modeling enterprise systems is relatively old and emerged in early 2000s (Luckham, 2002; Michelson, 2006) but gained popularity in the recent years by enablement through technological advancements, such as cloud computing and web protocols (Lan et al., 2015, Pages 4-5). Also Michelson has laid considerable groundwork in 2006 (Michelson, 2006) to the literature state of the topic. His “Event-driven Architecture Overview” is heavily referenced and cited.

Secondly the event-driven approach to software architecture correlates to SOA and Microservices (Levina and Stantchev, 2009, Page 39). Furthermore, the application fields lie in beneficiaries of real-time processing and analytics such as IoT, industry4.0 or anything automation related (Lan et al., 2015; Levina and Stantchev, 2009; Theorin et al., 2017).

The search string to find literature in Google Scholar and ProQuest as databases is composed of three parts:

1. The topic (e.g., event-driven and event-processing)
2. The application fields (industry4.0, IoT, real-time and automation)
3. Technical terms (loose coupling, programming framework and architecture)

The technical terms were selected to restrict the results to technical publications to guarantee an in-depth technical literature review. Since Cloud computing has gained huge popularity in recent years and Michelson has laid groundwork for EDA in 2006, the results were limited to being newer than 2004. Using LitSonar the search string for ProQuest was built as described above:

((AB(event-driven OR event-processing) OR TI(event-driven OR event-processing) OR SU(event-driven OR event-processing)) AND (AB(industry4.0 OR IoT OR real-time OR automation) OR TI(industry4.0 OR IoT OR real-time OR automation) OR SU(industry4.0 OR IoT OR real-time OR automation)) AND ((AB(architecture) OR TI(architecture) OR SU(architecture)) OR ((AB(loose) OR TI(loose) OR SU(loose)) AND (AB(coupling) OR TI(coupling) OR SU(coupling))) OR ((AB(programming) OR TI(programming) OR SU(programming)) AND (AB(framework) OR TI(framework) OR SU(framework))))) AND (YR(>=2005) AND YR(<=2022))

Respectively for Google Scholar:

allintext: (event-driven OR event-processing) AND (industry4.0 OR IoT OR real-time OR automation) AND ("loose coupling" AND architecture)

The whole process of finding the underlying literature is shown in figure 2. The initial results with the simple search term “Event-driven Architecture” were 3.89 million hits in Google Scholar and 26.565 hits in ProQuest. Obviously, these amounts of hits are too much to handle properly. So after screening the first few hits respectively to gain an idea of the topic, the previously described search strings were created based on the findings. This resulted in 3130 hits in Google Scholar and 122 hits in ProQuest. These amounts of literature were much better to handle and were screened for titles, abstracts, keywords and limited to English language. A sub-selection of five publications of Google Scholar and four publications of ProQuest was made and analyzed regarding their introduction and conclusion to confirm their relevance. These nine publications were then decided for and through appliance of a backward search, two further publications were considered. The forward search didn't provide relevant publications with new perspectives.

So ultimately, after applying above search strings, screening for results, and applying backwards and forward searches, the final number of reviewed papers is 12.

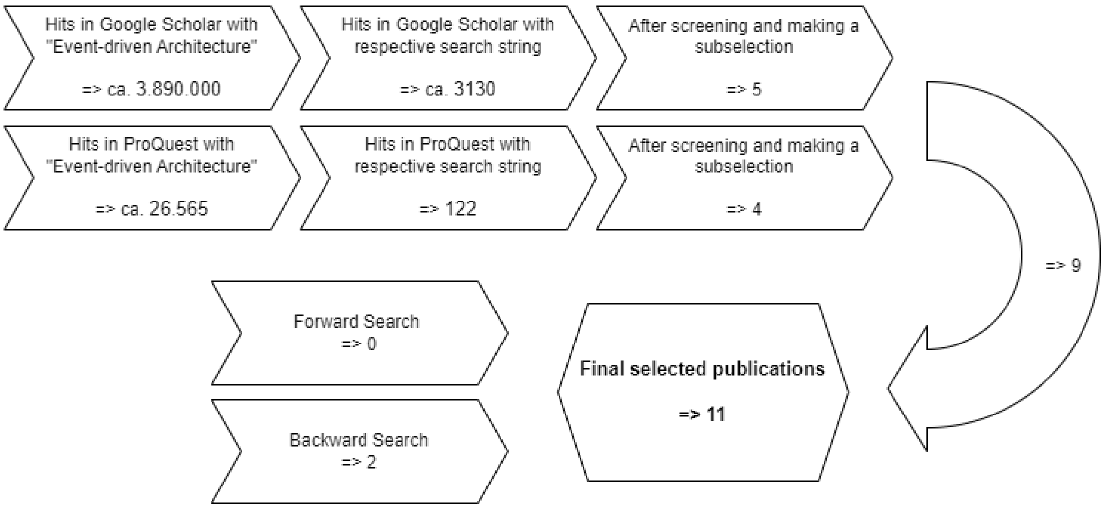
Hits in Google Scholar with "Event-driven Architecture" => ca. 3.890.000
Hits in Google Scholar with respective search string => ca. 3130
After screening and making a subselection => 5
Hits in ProQuest with "Event-driven Architecture" => ca. 26.565
Hits in ProQuest with respective search string => 122
After screening and making a subselection => 4
=> 9
Forward Search => 0
Backward Search => 2
Final selected publications => 11

Figure 2: Literature screening process

4 Results

To gain a concept-centric perspective based on the findings above, Webster & Watson recommend creating a concept matrix. By categorizing the underlying literature and synthesizing it into identified concepts, it helps the reader "to make sense of the accumulated knowledge on a topic" (Webster and Watson, 2002, Pages 17-18). The resulting concept matrix can be seen in Table 1. After the full text analysis of the selected 11 publications, the respective publications were classified by their contents into: (1) EDA overview, (2) Implementations of EDA, (3) CEP, (4) SOA, (5) IoT and (6) industry.

While EDA overview (1) describes publications that describe EDA in general in a paragraph or more, Implementations of EDA (2) indicates that the authors describe a proposed implementation of EDA. Both CEP (3) and SOA (4) reflect that a publication explains the respective term or shows an implementation with CEP and/or SOA being an integral part. IoT (5) and industry (6) are meant to classify the publications by their underlying context and application field.

Publication	Concept					
	EDA overview	Implementations of EDA	CEP	SOA	IoT	Industry
Michelson, 2006	x		x	x		
Levina and Stantchev, 2009	x		x	x		
State Grid Electric Power Research Institute et al., 2018		x	x			
Lan et al., 2015	x	x		x	x	
Theorin et al., 2017		x		x		x
Chiprianov et al., 2014	x	x				
Rieke et al. 2018		x		x		
Industrie 4.0 Working Group, 2013						x
Nitz et al., 2013		x	x		x	
Magid et al., 2008		x	x			
Sousa et al., 2021		x	x			

Table 1: Concept matrix

4.1 RQ1: What is Event-driven Architecture?

EDA consists of three main components: The event source, the event consumer and in between event processors which process incoming events and forward them accordingly (Levina and Stantchev, 2009, Page 38). The centerpiece of attention in this architectural approach is the event itself which is passed form the event generator to the event consumer (Michelson, 2006, Page 3). Event sources can be applications which raise a notable event within their business logic or sensors in an IoT context noticing an incident such as a threshold being exceeded (Levina and Stantchev, 2009, Page 38). In consequence, the event source or generator constructs the event and transmits it to the event channel or directly to the processor (Chiprianov et al., 2014, Page 360; Michelson, 2006, Page 6). Therefore, the event source is completely decoupled from the event's subsequent processing. This drastically reduces the number of dependencies (or direct references) to downstream processors to one: the event channel or processor to which the event is being published (Chiprianov et al., 2014, Page 360). In contrast, a classical PtP approach requires the event source to know about all interested parties and directly transmit the content of the event.

The event consumers are all interested subscribers, which perform their functionality upon receival of the event (Levina and Stantchev, 2009, Page 38). This is also referred to as “Downstream activity”. The consumers interest in raised events ranges from the content of a single event such as thresholds being exceeded (Michelson, 2006, Page 5) to the aggregate of events originating from an entire power grid (State Grid Electric Power Research Institute et al., 2018). Therefore, the downstream activity itself can also be anything from simple alerts being raised, display of status reports to real-time missile-detect-and-engage systems in a military context (Magid et al., 2008, Page 252). The complexity of downstream actions defines the necessary complexity of the systems middleware for event transmittal and processing. Simple downstream actions only necessitate a simple publish/subscribe mechanism in which the middleware can be described as an event channel which holds the event that was published and serves it to consumers (Michelson, 2006, Page 6). For complex downstream actions which base on an aggregate of events of different sources and/or times (Levina and Stantchev, 2009, Page 38), a more mature middleware is required which properly processes events and generates new and high-level aggregated information for the event consumers (Rieke et al., 2018, Page 11). This broader approach is described as complex event processing (Levina and Stantchev, 2009, Page 38).

In essence, EDA simply decouples event sources from event consumers and usually provides rich amounts of historical data from previous events (Michelson, 2006, Page 5). But the implications for the software landscape of a business are huge.

4.2 RQ2: What are the Challenges and Benefits of EDA?

EDA in all its variations offers a broad width of benefits and strengths just by its asynchronous and loosely coupled nature. Since the creator of an event is detached from any knowledge of the events subsequent processing (and usually vice versa), EDA systems are extremely loosely coupled and are thus “best used for asynchronous flows of work and information” (Michelson, 2006, Page 3). A major argument for asynchronous flows (for example achieved through a Publish/Subscribe pattern) is the option to store events before processing and therefore allowing a staggered processing of events. In a mobile computing scenario, “a mobile device has the possibility to retrieve events delivered while it was turned off” (Nitz et al., 2013, Page 60). On the contrary, EDA also provides an environment for (near) real-time applications (State Grid Electric Power Research Institute et al., 2018). Additionally EDA provides an excellent foundation for display of dashboards to give real-time insights on processes and performance and thus also allowing an almost real-time reaction (Levina and Stantchev, 2009, Page 38). Another purpose of a generated event might be to capture the state of an object/situation for historical purposes (Michelson, 2006, Page 5), providing a history of state changes for a context of a collection of events. For example, this might be an order going through several steps from issuing the order to delivering the products. Another strength of EDA roots in its decoupled nature with its many-to-many communication driven mainly by the events (Lan et al., 2015, Page 6): EDA systems are platform- and technology-agnostic with the only constraints being able to interpret the events and being able to publish/receive events through the middleware.

The major challenges and drawbacks of EDA stem in the previously explained strengths: complexity, development overhead and therefore cost. The biggest challenge lies in the implementation of EDA and the complexity of real-life scenarios. The heterogenous input data for events, the general amount of data being processed and the definition for the domains events all provide notable challenges for developers (Rieke et al., 2018, Pages 11-12). Applying an EDA approach to a broad system (for example an enterprise) results in a CEP approach, due to the complexity of large IT landscapes and its interoperability (Chiprianov et al., 2014, Page 357-360). But “CEP in real-time is not easy to implement and adhere to, largely due to its complexity and variety of data” (Sousa et al., 2021, Page 2).

4.3 RQ3: What are common approaches?

EDA can be approached in combination with different architectural patterns or by itself. But for any approach, EDA relies on a middleware holding and providing the given events (Michelson, 2006, Page 4). The common approach is a Publish/Subscribe communication pattern (Rieke et al., 2018, Page 3).

It is important to examine EDA and SOA as two distinct architectural patterns which “are peers and complements” (Michelson, 2006, Page 2). Since the two concepts work so well together, there is an inclination to view EDA as a branch of SOA (Michelson, 2006, Page 2). By decomposing systems into their functionalities, the resulting services technologically neutral (between each other), loosely coupled and encapsulate business functionality (Levina and Stantchev, 2009, Page 38). It therefore improves infrastructure efficiency of the underlying system (Theorin et al., 2017, Page 1298) as well as general code quality by creating subsystems organized by their business functionality and responsibility. Still there are drawbacks to this approach, namely the need for a central service orchestrator, which handles service discovery and invocation (Theorin et al., 2017, Page 1299), and its reliance on a Request/Response mechanism making it a mostly synchronous approach (Lan et al., 2015, Page 6). In contrast, EDA offers extreme loose coupling and an asynchronous structure (Lan et al., 2015, Page 6).

EDA enhances SOA from two perspectives: The invocation of services through events (consumers) and the creation of events through service invocation (event generator) (Michelson, 2006, Page 2). By invoking services through events (also known as SOA 2.0 or Event-driven SOA), the decoupled nature of EDA makes the need for a central service orchestrator less strict and loosens the coupling and knowledge of the services to each other. Services aren’t invoked directly but by certain events being raised in the EDA middleware, resulting in more of a service choreography than orchestration (Theorin et al., 2017, Page 1299). By placing event generation in services, the weakness of a synchronous request/response mechanism of SOA gets softened. In cases where the client or a particular service only need to start a downstream action, EDA (with its asynchronous approach) dissolves the need to wait for a response and lets the client or service execute continuously (Lan et al., 2015, Page 6). This benefit logically doesn’t apply if the client needs a response from a service (such as validation of provided login data).

Ultimately EDA and SOA complement each other very well by maintaining each other’s benefits. “Merging these concepts results in an enterprise architecture that is more flexible while being robust to changes. Its components are loosely coupled and can be accessed in any business situation” (Levina and Stantchev, 2009, Page 39). For a broader scope, the application of CEP goes well beyond SOA in which SOA and EDA aren’t complements any more but SOA is just the technological environment in which a CEP might be embedded (Michelson, 2006, Page 2).

4.4 RQ4: What traditional and opposing patterns exist?

In manufacturing, applications and devices are traditionally connected using a PtP basis with a Client/Server infrastructure. The constraint of a fully connected PtP network makes it hard to add functionality or systems (Theorin et al., 2017, Page 1298). In real-time systems, the need for quick response times usually results in monolithic applications tightly coupled to the underlying hardware (Magid et al., 2008, Pages 251-252). With these traditional systems having their drawbacks with increasing amounts of devices and applications (Theorin et al., 2017, Page 1298) or simply developing cost (Magid et al., 2008, Page 252), modern systems need to adapt to keep up with a dynamic business environment, increasing amounts of IoT devices and data. The mentioned strengths of EDA seem to justify a departure from traditional patterns to implementations of EDA, mostly with CEP, (Chiprianov et al., 2014; Nitz et al., 2013; Theorin et al., 2017) or an incorporation of EDA to other architectural approaches such as SOA (Lan et al., 2015; Levina and Stantchev, 2009).

5 Discussion

After reviewing the literature, modern software seems to necessitate EDA. Either in form of a direct implementation, an evolution, or a deviation of it. The described strengths solve current and upcoming challenges software applications face, such as an abundance of data, devices and processes which need to be intertwined and provide a bigger picture for stakeholders. Furthermore, the natural scalability of EDA based systems suggests that businesses should take an EDA approach early on. The drawbacks of EDA may weigh heavy but can either be softened (complexity by encapsulation through SOA) or made obsolete entirely by incorporating EDA in other architectural approaches and creating synergy effects.

This literature review achieved to give an insight into EDA, common approaches and an examination of the benefits and challenges of EDA. Due to a cloudy and ambiguous definition of the space between event generators and event consumers in literature, the explanation of the event processor / event channel, is not clear. A clear distinction of the terminology used (Event processor, event channel, event sink, event processing engine, etc.) is necessary to better understand this vital part of EDA. As can be seen in the concept matrix (Table 1), this work failed to find sufficient literature which can be classified by IoT or industry specific. Therefore, a clear insight on the current situation of EDA in industry couldn’t be given. Most publications scratching these classifications were proposals of implementations with EDA but no proper analysis of businesses and manufacturers and their current software landscape and spread of EDA.

Furthermore, the current literature lacks a deeper examination of EDAs limitations and costs. EDA as a more sophisticated approach than traditional software architecture needs more skilled developers. Events need to be stored and a publish/subscribe middleware needs to be hosted and constantly available. Therefore future work needs to examine the costs of EDA in comparison to currently widespread traditional architectures. How expensive is the storage and transmittal of events in a cloud computing environment compared to an on-premise solution? How much training is needed for developers to incorporate the EDA philosophy?

The literature paints a clear picture that investments in EDA implementations are reasonable, but it is especially important to examine the return of investment on different variations EDA implementation. Since industries are reluctant to integrate new technologies (Theorin et al., 2017, Page 1297), such an analysis helps to inspire businesses to embrace EDA.

6 Conclusion

Adoptions of EDA solve current and upcoming challenges for businesses and software systems in general. Large amounts of data become manageable, the high degree of coupling creates maintainable subsystems and systems become more robust through the asynchronous nature of EDA. By these benefits alone, previously untaken possibilities become available. Therefore, enabling EDA seems to be beneficial for future software projects. But the current literature lacks metrics and in-depth analyses of costs and profits to help stakeholders in deciding for or against the overhead of implementing EDA. Furthermore, CEP is a very powerful approach to EDA but needs skilled developers and assumes great knowledge of the domain since it relies on a large variety of data. The research questions have been mainly answered, even though the current industry situation of EDA implementation couldn’t be analyzed thoroughly.

References

Chen, C.Y., Fu, J.H., Sung, T., Wang, P.-F., Jou, E., Feng, M.-W., 2014. Complex event processing for the Internet of Things and its applications, in: 2014 IEEE International Conference on Automation Science and Engineering (CASE). Presented at the 2014 IEEE International Conference on Automation Science and Engineering (CASE), pp. 1144–1149. https://doi.org/10.1109/CoASE.2014.6899470

Chiprianov, V., Falkner, K., Szabo, C., Puddy, G., 2014. Architectural Support for Model-Driven Performance Prediction of Distributed Real-Time Embedded Systems of Systems, in: Avgeriou, P., Zdun, U. (Eds.), Software Architecture, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 357–364. https://doi.org/10.1007/978-3-319-09970-5_30

Laigner, R., Kalinowski, M., Diniz, P., Barros, L., Cassino, C., Lemos, M., Arruda, D., Lifschitz, S., 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). Presented at the 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 213–220. https://doi.org/10.1109/SEAA51224.2020.00045

Laliwala, Z., Chaudhary, S., 2008. Event-driven Service-Oriented Architecture, in: 2008 International Conference on Service Systems and Service Management. Presented at the 2008 International Conference on Service Systems and Service Management, pp. 1–6. https://doi.org/10.1109/ICSSSM.2008.4598452

Lan, L., Wang, B., Zhang, L., Shi, R., Li, F., 2015. An Event-driven Service-oriented Architecture for Internet of Things Service Execution. Int. J. Online Eng. IJOE 11. https://doi.org/10.3991/ijoe.v11i2.3842

Levina, O., Stantchev, V., 2009. Realizing Event-Driven SOA, in: 2009 Fourth International Conference on Internet and Web Applications and Services. Presented at the 2009 Fourth International Conference on Internet and Web Applications and Services, pp. 37–42. https://doi.org/10.1109/ICIW.2009.14

Luckham, D., 2002. The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Professional.

Magid, Y., Oren, D., Botzer, D., Adi, A., Shulman, B., Rabinovich, E., Barnea, M., 2008. Generating real-time complex event-processing applications. IBM Syst. J. 47, 251–263. https://doi.org/10.1147/sj.472.0251

Michelson, B., 2006. Event-Driven Architecture Overview (No. 681). Patricia Seybold Group, Boston, MA. https://doi.org/10.1571/bda2-2-06cc

Nitz, S., Kleiner, C., Koschel, A., Astrova, I., 2013. Applying event-driven architecture to mobile computing, in: IEEE International Symposium on Signal Processing and Information Technology. Presented at the IEEE International Symposium on Signal Processing and Information Technology, pp. 000058–000063. https://doi.org/10.1109/ISSPIT.2013.6781854

Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0. Acatech, 2013.

Rieke, M., Bigagli, L., Herle, S., Jirka, S., Kotsev, A., Liebig, T., Malewski, C., Paschke, T., Stasch, C., 2018. Geospatial IoT—The Need for Event-Driven Architectures in Contemporary Spatial Data Infrastructures. ISPRS Int. J. Geo-Inf. 7, 385. https://doi.org/10.3390/ijgi7100385

Sousa, R., Miranda, R., Moreira, A., Alves, C., Lori, N., Machado, J., 2021. Software Tools for Conducting Real-Time Information Processing and Visualization in Industry: An Up-to-Date Review. Appl. Sci. 11, 4800. https://doi.org/10.3390/app11114800

State Grid Electric Power Research Institute, Zhou, M., Yan, J., State Grid Electric Power Research Institute, 2018. A new solution architecture for online power system analysis. CSEE J. Power Energy Syst. 4, 250–256. https://doi.org/10.17775/CSEEJPES.2017.00430

Theorin, A., Bengtsson, K., Provost, J., Lieder, M., Johnsson, C., Lundholm, T., Lennartson, B., 2017. An event-driven manufacturing information system architecture for Industry 4.0. Int. J. Prod. Res. 55, 1297–1311. https://doi.org/10.1080/00207543.2016.1201604

Venkatesan, D., Sridhar, S., 2017. A novel programming framework for architecting next generation enterprise scale information systems. Inf. Syst. E-Bus. Manag. 15, 489–534. https://doi.org/10.1007/s10257-016-0330-y

Webster, J., Watson, R.T., 2002. Analyzing the Past to Prepare for the Future: Writing a Literature Review. MIS Q. 26, xiii–xxiii.

Zhang, Y., Duan, L., Chen, J.L., 2014. Event-Driven SOA for IoT Services, in: 2014 IEEE International Conference on Services Computing. Presented at the 2014 IEEE International Conference on Services Computing, pp. 629–636. https://doi.org/10.1109/SCC.2014.88