24th International Conference on Business Information Systems (BIS 2021)

Big Data

https://doi.org/10.52825/bis.v1i.67

© Authors. This work is licensed under a Creative Commons Attribution 4.0 International License

Published: 02 July 2021

Towards a Concept for Building a Big Data Architecture with Microservices

Aamir Shakir, Daniel Staegemann, Matthias Volk, Naoum Jamous and Klaus Turowski

Otto-von-Guericke University Magdeburg

Abstract. Microservices and Big Data are renowned hot topics in computer science that have gained a lot of hype. While the use of microservices is an approach that is used in modern software development to increase flexibility, Big Data allows organizations to turn today's information deluge into valuable insights. Many of those Big Data architectures have rather monolithic elements. However, a new trend arises in which monolithic architectures are replaced with more modularized ones, such as microservices. This transformation provides the benefits from microservices such as modularity, evolutionary design and extensibility while maintaining the old monolithic product's functionality. This is also valid for Big Data architectures. To facilitate the success of this transformation, there are certain beneficial factors. In this paper, those aspects will be presented and the transformation of an exemplary Big Data architecture with somewhat monolithic elements into a microservice favoured one is outlined.

Keywords: Big Data, Microservice, Success Factors, Software Design, Software Architecture

1 Introduction

A continuous growth of the overall amount of data that is created, captured and analyzed heavily affects today's infrastructures as a whole [1]. With these advancements, an increase in processing speed is desirable. Consequently, the performance of conventional methods and technologies are growing insufficient, thus motivating the invention of new techniques and the need of modern data architectures, as they are denoted by the terms Big Data and Big Data architecture [2]. Organizations that implement aforementioned technologies can experience a significant increase in performance [3] due to more efficient decision making, reduced expenses and a more result-oriented portfolio of services while improving customer relations [4]. However, implementing the respective applications is still a challenging task [4]. Contributing to this problem, the newly implemented algorithms and solutions might evolve over time, thus rendering the effort pointless, if the Big Data architecture is not adapted accordingly. This results in a rising demand for designs, which consist of building blocks that can be modified and replaced independently from each other, contrasting isolated implementations and applications [5]. Implementing microservices as the building blocks of Big Data architecture appears to be a feasible solution due to high degree of modularization [6]. Therefore, the research question is:

What Factors should be considered to build a microservice favored Big Data architecture?

To provide an answer, at first, the concepts of microservices and Big Data are described. This is followed by a look at what research has already done in the area of microservices in Big Data. This shall be a foundation to derive success factors for a successful microservice based

Big Data architecture. Afterwards, an exemplary Big Data architecture with isolated elements will be transformed into a microservice favored Big Data architecture. he advantages and disadvantages are discussed and compared with the specified success factors. Finally, a conclusion is given, also highlighting potentially beneficial directions for future research endeavors.

2 Background

Big Data and microservices are introduced in the following. Their mode of operation and important properties are described. Also monolithic architectures as alternatives to microservice based systems are described and a state of the art of microservices in Big Data is given.

2.1 Big Data

While initially being referred to as a synonym for large amounts of data that cannot be easily handled by relational databases and technologies of that time, today Big Data covers a variety of advanced data characteristics, technologies, paradigms and methods [1]. During this time, the concept has undergone significant changes that dramatically changed the term from a hype topic [7] to the foundation of most of the data-driven and data-intensive projects known today [4]. Despite that long lasting maturation and a highly active research community [8], no distinct and universally applied definition was found that precisely describes the nature and elements of that term [1]. Notwithstanding that, according to one of the most widely used definitions, Big Data "consists of extensive data sets -primarily in the characteristics of volume, variety, velocity, and/or variability -that require a scalable architecture for efficient storage, manipulation, and analysis" [9]. Another definition which can be used is that Big Data is a field that systematically deals, extracts information of, or finds ways to analyze data volumes that are for example too complex, too fast-moving, too large or too weakly structured to be analyzed using manual and conventional data processing methods [10]. Similar to the pure definition itself, many differences about the description of the data exist. While some of the data characteristics are observed as core characteristics, namely volume, variety and velocity, others are treated unequally [11]. Volume refers to the size and number of elements to be processed, the variety focuses on the structure of data, which can be either unstructured, semi-structured or structured. Furthermore, the velocity describes the speed the data is incoming and processed with [12]. Apart from the pure lack of experts and qualified staff [13], the comprehensive planning, engineering and integration of architectures represents a cumbersome task [1]. Many practitioners and researchers noted this problem and attempted to reduce the prevailing complexity through the design and developments of promising solutions, such as reference architectures [14], decision support systems [4], automation approaches [15] or the application of new technologies [6]. Especially in times at which highly decentralized or loosely coupled environments are sought more than ever, as in the case of very large business application scenarios, the use of Big Data in combination with the latter remains desirable.

2.2 Monolithic Architectures

Monolithic architectures follow a traditional model for software in which the structure is a single and indivisible unit [16]. A monolith has one code base with multiple modules, like the described Big Data architecture in Section 5. Those modules can consist of one or multiple layers [17]. Figure 1 shows an example with three layers [18]:

I Front-end user interface that runs on user devices
II Logic components that run on a server
III Back-end database in which the application data is stored

Therefore, the modularization of such systems is limited by the resources they share (i.e, the database) and the components cannot be executed independently [19]. The central component and control can lead to a large code base, which makes the code difficult to understand and modify as well as less expandable [20]. As a result, the development slows down. Another effect is that continuous deployment can become difficult [21], since a small change to a part of the application requires the entire monolith to be rebuilt and deployed [21]. Big Data technologies are often very specific, and so is the connection between them. Special adapters, interfaces or services must be provided so that they are compatible. A ”loosening” of the system and flexible exchange is only possible to a very limited extent, if at all. Accordingly, such a system can appear monolithic to the user, whereas the conceptual design and implementation, on closer inspection, appear rather isolated and less flexible.

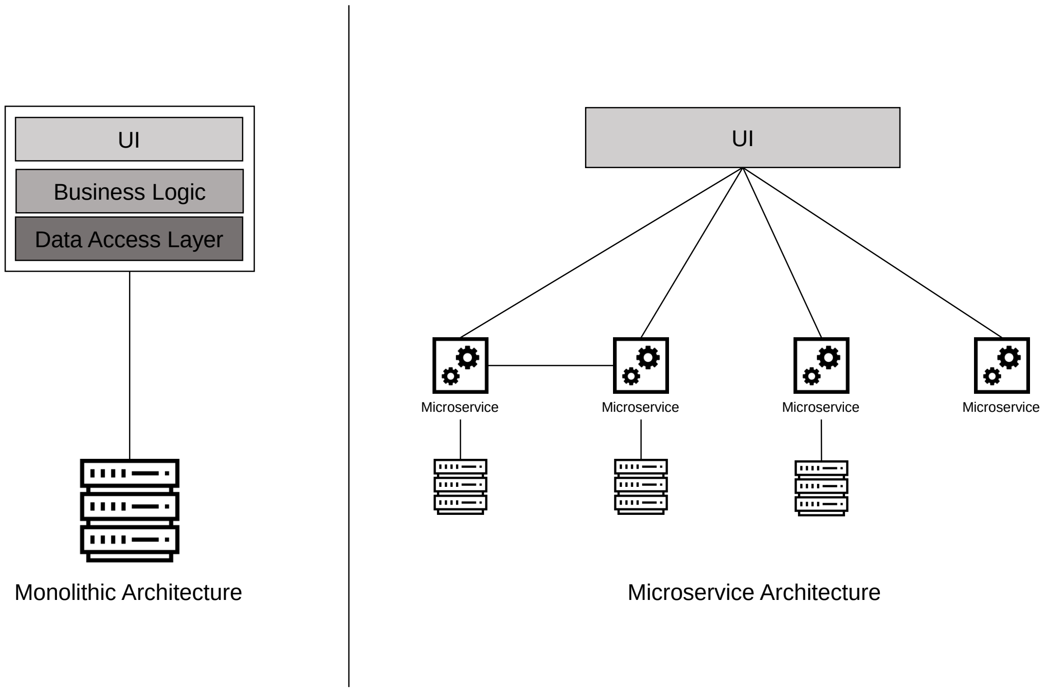

Figure 1. On the left side there is a monolithic architecture which contains all features and services versus on the right side the microservices architecture with decomposed services that work together to get the same functionality.

2.3 Microservices

Microservices are an opposite approach to monolithic architectures [17]. While there is no precise definition for the term microservice [22], they can be described as an approach to develop a single application by decomposing it into a suite of small services [23]. Each of them runs in their own process and communicates with lightweight mechanisms. These services, because being so independent, only need a minimum of central management [24]. They are based on business functions and can be deployed separately by continuous deployment tools. Due to their independence, they can be written in different programming languages and can be based on different technologies, for example for the data storage. Common characteristics for microservices are described in the following. The first one is *”Organized around Business Capabilities”*. The microservice approach leads to structuring teams around business capabilities instead of traditionally building teams based on the technology layer. Consequently, the teams are cross-functional and encompass the full range of skills required for development and prevent the “logic everywhere” siloed architectures [22]. This comes with the constraint that the team realizing this concept can’t be based on strict hierarchical communication [25]. The second one is *”Componentization, or Modularity”*. Componentization is known to be a generally good practice in software engineering. Achieving a high degree of modularity is often considered as difficult [26]. Because systems are broken down into services that are independently deployable, componentization is achieved with the microservice architecture by design. For small internal changes, only the affected service has to be redeployed. The third one is *”Decentralized Data Management”*. Every microservice has its own storage and its own technology to manage its store. This leads to a decentralized data storage which is isolated from other services. Different services can therefore have different conceptual models, e.g. they operate on different attributes of the same data entities. The fourth one is *"Evolutionary Design"*. The evolutionary design is a typical feature of a microservice architecture, where services decomposition is used as a driving force to enable frequent and controlled changes in the system [27]. The microservices are specialized, so minor changes on the feature request can lead to implementing new services which can be easily added to the existing application. However, in case of only small changes, those can usually be based on their predecessor. The fifth one is *"Smart endpoints and dumb pipes"* [22]. The services run isolated from each other. They are decoupled, cohesive and have their own domain logic. They use lightweight protocols for communication, which are often used in a REST-ish manner and as basic as possible. The sixth one is *"Products not Projects"*. The aim of most application developments is to deliver a piece of software which is considered to be complete. While developing a service, the team preferably develops its service as a standalone product. The seventh one is *"Decentralized Governance"*. Because of the decentralized architecture, which relies on independently deployable components, the centralized governance can be relaxed [28]. Every service in the system can be build based on its own technology that is most suitable for the task. This leads to high flexibility in the choice of tools and implementation technology. Additionally it allows us to adopt to new technologies on a smaller scale first [23]. The properties are summarized in Table 1.

Table 1. An overview of the typical properties of microservices.

Property	**Description**
Organized around Business Capabilities	Developer teams are structured around Business Capabilities instead of the technology layer.
Componentization or Modularity	The system is broken down into services that are independently deployable.
Decentralized Data Management	Every service has its own storage and its own technology to manage its store.
Evolutionary Design	For each service the best fitting technology (e.g. programming language, framework) can be used.
Smart Endpoints and Dumb Pipes	Components need to be able to access required data and other components through pipelines [22].
Products not Projects	The microservice is supposed to be a finished standalone product.
Decentralized Governance	Every service in the system can be build based on its own technology that is most suitable for the task.

3 Microservices in Big Data

There are several contributions for example [27]–[29], which explore to what extent Big Data can be used in microservice architectures. This is generally considered in the context of IoT. Furthermore, some approaches also use microservices to build a Big Data architecture. Zhelev and Rozeva showed that microservices can be used to build an event-driven Big Data architecture that relies heavily on the implementation of microservices [30]. Also Miao et al. implemented a microservice based Big Data Analysis Platform for Online Educational Applications [31]. Both describe how they build their architecture and they also described that the main challenge is to keep the data integrity. Freymann et al. described how modular services such as microservices can be used to tackle the six fundamental challenges: *Volume*, *Velocity*, *Variety*, *Complexity*, *Veracity* and *Value* [6]. They found that the modular architecture should have the properties described in Table 2. Like Zhelev and Miao, they outlined that data integrity is a challenge but really important for a successful Big Data architecture. Furthermore, the use of microservices for the application of test-driven development to the Big Data domain is proposed in the work of Staegemann et al. [32].

Table 2. In the table the properties of the modular architecture are described, which was used to build a Big Data architecture [6].

Property	**Description**
Modularity	The architecture divides and structures a system into software and hardware modules realized by microservices.
Adaptability	This is the ability of the architecture to modify and extend a system [33].
Scalability	Scalability supports the expansion of a solution horizontally and vertically by its hardware and software components [6].
Data Handling	A proper deal handling is important to organize large amount of data.
Distributed System	Distributed Systems ”enable load balancing, distribution of computational power, data storage and efficient parallel” [6].
Infrastructure Management	An overall management of the system to get transparency [34].

4 Success Factors of Microservices in Big Data

Based on the findings from Section 3 and the definitions and characteristics of Big Data and microservices, certain factors can be identified that facilitate a successful implementation of microservice favored Big Data architectures. The projects described in Section 3 have taken data integrity as a challenge in implementing their architecture. In many Big Data architectures, a structured and segregated system is created, to prevent unplanned modifications during the data processing, transformation or persistence. With the modular structure, there is no such instance. Every process can hold its own data and change it. In turn, this change can be relevant for other processes. If these changes are not communicated further, this can lead to serious errors and failure. Therefore, the first success factor for microservice favored Big Data architectures is maintaining data integrity. Many projects using microservices for Big Data systems have tried to apply best practices. But since their systems often require a high degree of flexibility and specialization, the implementation was massively facilitated by loosening them a bit. This effect has also been described by Krylovskiy et al. for building large-scale Smart City IoT platforms [27]. A microservice architecture with Big Data is implemented successfully by deviating from best practices for the architectures. This shows that it isn't mandatory to always follow strict rules and that it is also possible to deviate from them if it makes the implementation easier. This is the second success factor. Based on the findings described in Section 3 and the characteristics of microservices described in Section 2.3, further important factors can be extracted. One of them is that microservices benefit greatly from their extensibility. This is also highlighted in the work of Freymann et al. [6]. New components can be added without further ado. This makes it possible to swap out components with relative ease. Furthermore, it is easy to scale the system horizontally and vertically [28]. Another success factor it the technological independence of the individual components. This also allows to use the most suitable technology for each component, instead of being forced to adhere to predefined standards. For example, [35] describes a fraud detection system that uses different databases - one for the user's past activity, another for a blacklist, a third one for a white list of activities etc. Each of these services can follow a different architecture and a different internal mechanism. Lastly, the individual independent components must communicate in a way among themselves that fits the architecture they are following. When dealing with microservices, this is done via pipelines, which follow the ”smart endpoints and dumb pipes” approach. That means, the components should not aim at forming fine-grained and complex communication structures with other components. Instead, they should try to be ”as decoupled and as cohesive as possible” [22]. Those factors are also summarized in Table 3.

Table 3. The success factors for building a microservice favored Big Data architecture are described in this table.

Success Factor	**Description**
Data Integrity	The data has to be consistent through every process and every change.
Don’t be a Slave of Rules	Best practices can be relaxed if it is deemed beneficial to build a successful architecture.
Extensibility	To refine or update an existing architecture, new components are to be added easily [22].
Technological Independence	For each service the best fitting technology (e.g. programming language, framework) can be used.
Pipelines	Components need to be able to access required data and other components through pipelines [22].

5 Example Transformation

In the previous section, the success factors were explained. In the following, a Big Data architecture is transformed into a microservice favored architecture. There are many different Big Data architectures, generic ones such as the Lambda architecture [36], the Kappa architecture [37] as well as domain-specific ones as in the example of the IoT Big Data Architecture proposed in [38]. In the following, a widely used initial architecture is used for the transformation [38]. This can be used as an example to show how one should also proceed with other systems. Due to its already modular structure, there are few hurdles to overcome. As will be shown in the following, this architecture benefits greatly from the transformation. The architecture is presented first, followed by an explanation of the transformation. The advantages and challenges are then discussed and the success factors presented in Section 4 are examined.

5.1 Exemplary Big Data Architecture

The fundamental building blocks of Big Data architectures are *data sources*, *batch processing*, *real time message ingestion*, *analytical data storage*, *analysis and reporting* and *orchestration*, as they are visualized in Figure 2.

To start with Big Data applications, one or multiple sources of data are required. The specific kind of sources may vary and may, inter alia, involve application data from databases, files produced by the application itself (e.g. log files) or real-time data from sensors or other IoT devices. Before an analysis can be performed, the data has to be processed first. The initial filtering, aggregation and further preparation for analysis is usually conducted via *batch processing* [38]. This process of reading and writing source files to a new destination after filtering them takes a long time, depending on the quantity of data. If the data set consists of real-time messages, the application has to provide a way to handle them. Such real-time messages are received in the form of a stream, from which the data can for example be saved by simply dropping the messages into a folder or analyzed in real-time, with only the results being stored. To support scale-out processing, reliable delivery and other queueing semantics, an ingestion store as a buffer may be needed. Once preparations are finished, Big Data applications usually store the data in a structured format, which in turn can be efficiently used by analysis tools. To further the aspect of efficiency, the components mentioned above have to be connected and applied repeatedly on the data. Technologies that aid the automation of these workflows are called *orchestration technologies* [38].

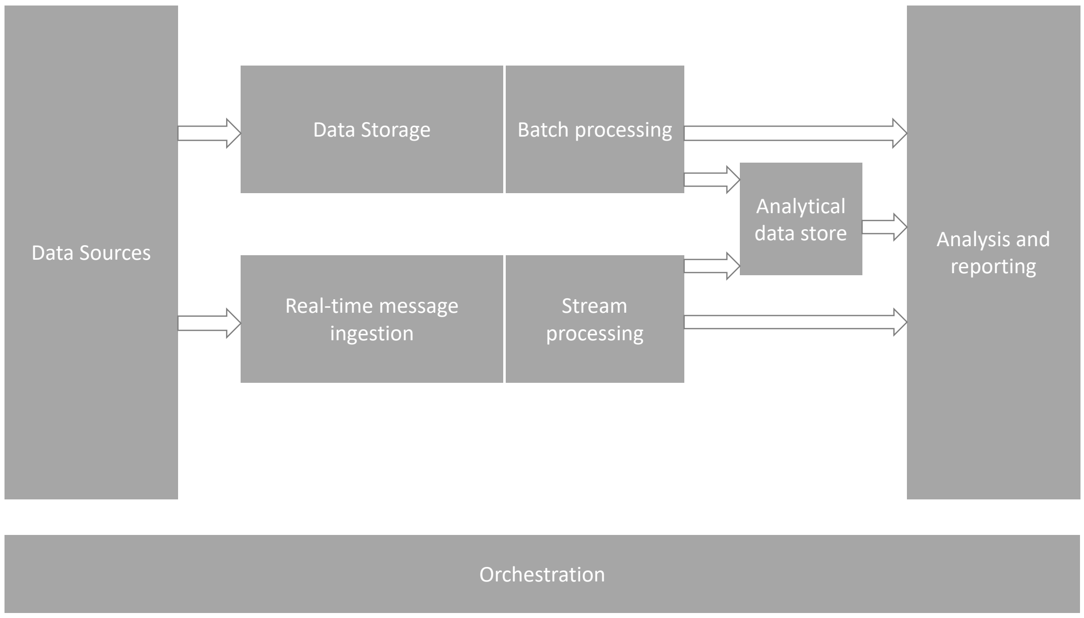

Figure 2. The structure of a Big Data architecture. It contains the components described above and also the dependencies between them. (As shown in [38].)

5.2 Transformation

The central control of the routines in the orchestration is closely interwoven in the presented architecture. Therefore, slightly simplified, it can be abstracted as one element. In the first step, this module is removed. For the repeating routines, an extra module can be written in each case, which accesses the required components through interfaces. Thus the functionality is maintained, but at the same time one moves away from closely intertwined constructs. Since the described architecture already possesses a modular structure, the individual components can be transferred into their own services. The following must be considered with the transfer:

- Dependencies: Required services are addressed via pipelines.
- Interfaces: Other services can use the results, which must be provided by interfaces.

If each service ensures these points, a coherent transformation can take place. Corresponding services can address other services via pipelines and interfaces. This means, that the dependencies can be transferred from the old model without further ado. So the previously thought-out processing steps are retained, but at the same time the rigid structure is broken. By its very nature, a Big Data architecture is usually dealing with very large amounts of data. The *data source* is a separate service. However, it must be ensured in any case that, if a service wants to make constant changes to a certain part of the dataset, this is also communicated to the *data service*. This is necessary to ensure the consistency of the data, one of the success factors mentioned in Section 4. Likewise, from a performance point of view, it makes a lot of sense for certain services to keep a part of the data with them as well. At this point, the best practice of microservices, which states that each service owns its own data, is relaxed. This is in accordance with the second success factor (”Don’t be a slave of rules”). However, this is not possible with the Big Data architecture, since the data cannot always be clearly assigned to each service. It therefore makes sense for a service to cache certain data, but for the entire dataset to be managed by an extra service. In further research it should be investigated in

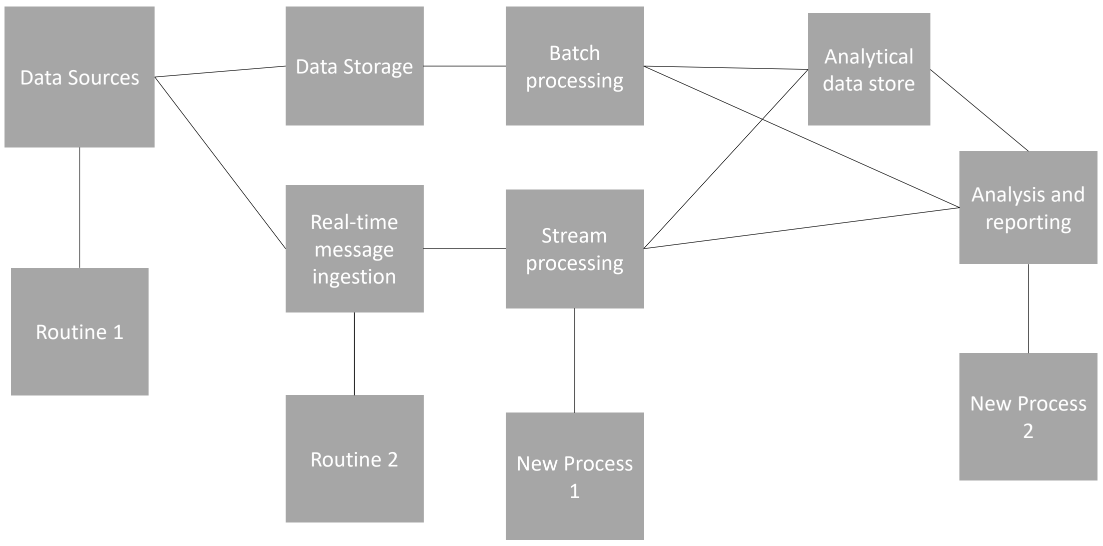

Figure 3. The result of the transformation. The whole structure is significantly more modular, but still contains all components except for the orchestration. In addition, two services have been added as examples, each of which shows a routine that would otherwise find a place in the orchestration.

which dimension the overhead and the performance change it. Up to this point no clear statement can be made. Thanks to the use of interfaces, we can now extend our structure easily by new components as are exemplary shown in Figure 3. By this, our architecture fulfills another of the success factors from Section 4. Figure 3 also shows the other components the now microservice-based architecture resulting from the transformation that has been undertaken.

5.3 Advantages and Challenges

The architecture shown combines a lot of positive aspects from both worlds. That means, the advantages of the individual mother architectures can also be transferred here. By decomposing the individual components into services, the ideal technology (e.g., specific algorithms or programming languages) can be used for each service. This architecture is also extensible. New services can be added without further ado. Via pipelines and appropriate interfaces they can request the processing necessary data of the respective services. With extensibility comes scalability. Big Data architectures support horizontal scaling [39]. Since the functionality was split into microservices, with the dependencies and components being adopted, not only the architecture is horizontally and vertically scalable, but also distinct parts of it can be adjusted to the respective circumstances and needs [40]. This makes the microservice favored architecture just as powerful as the reference architecture but at the same time brings the advantages of microservice architectures: Technological Independence, Evolutionary Design and Extensibility [41]. Big Data architectures are already complex from scratch. Although the transfer to a microservice architecture removes some of the complexity, new complexity is added from another side. This is because organizing the interfaces is complex during the initial transition. Another problem is the overhead of maintaining data consistency.

5.4 Evaluation of the Transformation in View of the Success Factors

In the following, we will compare the success factors presented in Section 4 with the architecture presented. During the transformation, effort is taken to set pipelines in such a way that, if a service wants to make consistent changes to the data, it must communicate this to the service managing the data, data sources, and this assumes the change. Likewise, a service compares its data with the data of the data service. This fulfills the point of data consistency and thus the first success factor. The communication between the individual services is very much based on pipelines, which are only available for data transfer. The data is in turn made available by well thought-out interfaces. This principle of smart endpoints and dumb pipes is fulfilled and thus also the second success factor, *Pipelines*. As already shown, new services can be added without further ado, so the third success factor - *Extensibility* - is also fulfilled. By decoupling them into individual services, they can be implemented in technologies that are best suited to them. That in turn is the next success factor, *Technology Independence*. During the transformation it was shown that the best practices were relaxed to ensure the most efficient and functional transfer possible. Therefore, the last success factor *Don't be a slave of rules* has also been fulfilled.

6 Conclusion

This paper showed which factors should be fulfilled in order to successfully build a microservice favored Big Data achitecture. Furthermore, an exemplary Big Data architecture was transformed step by step into a microservice based architecture on the conceptual level and examined with regard to the success factors. Thus, it was shown that this approach offers great potential in theory. In future research, the practical implementation should be examined in more detail. From a theoretical point of view, questions of data management and synchronization are still interesting. This would also be a question to be investigated in future research. In general, microservices offer a great perspective in Big Data and should be focused on more in the future.

References

[1] M. Volk, D. Staegemann, and K. Turowski, "Big Data," in *Handbuch Digitale Wirtschaft*, ser. Springer Reference Wirtschaft, T. Kollmann, Ed., Wiesbaden: Springer Fachmedien, 2020, pp. 1–18, ISBN: 978-3-658-17345-6. DOI: 10.1007/978-3-658-17345-6_71-1.

[2] F. X. Diebold, "On the Origin(s) and Development of the Term 'Big Data'," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2152421, Sep. 2012. DOI: 10.2139/ssrn.2152421.

[3] O. Müller, M. Fay, and J. vom Brocke, "The Effect of Big Data and Analytics on Firm Performance: An Econometric Analysis Considering Industry Characteristics," *Journal of Management Information Systems*, vol. 35, no. 2, pp. 488–509, Apr. 2018, ISSN: 0742-1222, 1557-928X. DOI: 10.1080/07421222.2018.1451955.

[4] M. Volk., D. Staegemann., S. Bosse., R. Häusler., and K. Turowski., "Approaching the (big) data science engineering process," in *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security - Volume 1: IoTBDS,*, INSTICC, SciTePress, 2020, pp. 428–435. DOI: 10.5220/0009569804280435.

[5] D. Staegemann, M. Volk, C. Daase, and K. Turowski, "Discussing relations between dynamic business environments and big data analytics," *Complex Syst. Informatics Model. Q.*, vol. 23, pp. 58–82, 2020. DOI: 10.7250/csimq.2020-23.05. [Online]. Available: https://doi.org/10.7250/csimq.2020-23.05.

[6] A. Freymann, F. Maier, K. Schaefer, and T. Böhnel, "Tackling the Six Fundamental Challenges of Big Data in Research Projects by Utilizing a Scalable and Modular Architecture," in *5th International Conference on Internet of Things, Big Data and Security, IoTBDS 2020. Proceedings*, 2020, pp. 249–256, ISBN: 978-989-758-426-8.

[7] L. Hung and F. Jackie, *Hype Cycle for Emerging Technologies, 2012*. [Online]. Available: https://www.gartner.com/en/documents/2100915/hype-cycle-for-emerging-technologies-2012 (visited on 02/14/2021).

[8] A. Parlina, K. Ramli, and H. Murfi, “Theme Mapping and Bibliometrics Analysis of One Decade of Big Data Research in the Scopus Database,” *Information*, vol. 11, no. 2, p. 69, Feb. 2020, Number: 2 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/info11020069.

[9] W. L. Chang and N. Grady, “NIST Big Data Interoperability Framework: Volume 1, Definitions,” Oct. 2019, Last Modified: 2020-01-07. [Online]. Available: https://www.nist.gov/publications/nist-big-data-interoperability-framework-volume-1-definitions.

[10] T. Breur, “Statistical Power Analysis and the contemporary “crisis” in social sciences,” *Journal of Marketing Analytics*, vol. 4, no. 2, pp. 61–65, Jul. 2016, ISSN: 2050-3326. DOI: 10.1057/s41270-016-0001-3.

[11] N. Khan, M. Alsaqer, H. Shah, G. Badsha, A. A. Abbasi, and S. Salehian, “The 10 Vs, Issues and Challenges of Big Data,” in *Proceedings of the 2018 International Conference on Big Data and Education*, ser. ICBDE ’18, New York, NY, USA: Association for Computing Machinery, Mar. 2018, pp. 52–56, ISBN: 978-1-4503-6358-7. DOI: 10.1145/3206157.3206166.

[12] A. Gandomi and M. Haider, “Beyond the hype: Big data concepts, methods, and analytics,” *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, Apr. 2015, ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2014.10.007.

[13] A. Gardiner, C. Aasheim, P. Rutner, and S. Williams, “Skill Requirements in Big Data: A Content Analysis of Job Advertisements,” *Journal of Computer Information Systems*, vol. 58, no. 4, pp. 374–384, Oct. 2018, ISSN: 0887-4417. DOI: 10.1080/08874417.2017.1289354.

[14] C. Avci, B. Tekinerdogan, and I. Athanasiadis, “Software architectures for big data: A systematic literature review,” *Big Data Analytics*, vol. 5, Aug. 2020. DOI: 10.1186/s41044-020-00045-1.

[15] A. M. Fernández, D. Gutiérrez-Avilés, A. Troncoso, and F. Martínez–Álvarez, “Automated Deployment of a Spark Cluster with Machine Learning Algorithm Integration,” *Big Data Research*, vol. 19-20, p. 100 135, Mar. 2020, ISSN: 2214-5796. DOI: 10.1016/j.bdr.2020.100135.

[16] A. Tiwana, “Chapter 5 - Platform Architecture,” in *Platform Ecosystems*, A. Tiwana, Ed., Boston: Morgan Kaufmann, Jan. 2014, pp. 73–116. DOI: 10.1016/B978-0-12-408066-9.00005-9.

[17] D. Namiot and M. Sneps-Sneppe, “On micro-services architecture,” *International Journal of Open Information Technologies*, vol. 2, p. 4, 2014.

[18] C. Fan and S. Ma, “Migrating monolithic mobile application to microservice architecture: An experiment report,” in *2017 IEEE International Conference on AI Mobile Services (AIMS)*, Jun. 2017, pp. 109–112. DOI: 10.1109/AIMS.2017.23.

[19] A. Bucchiarone, N. Dragoni, S. Dustdar, S. Larsen, and M. Mazzara, “From monolithic to microservices: An experience report from the banking domain,” *IEEE Software*, vol. 35, pp. 50–55, May 2018. DOI: 10.1109/MS.2018.2141026.

[20] P. Karwatka, *Monolithic architecture vs microservices*, Jan. 2020. [Online]. Available: https://divante.com/blog/monolithic-architecture-vs-microservices/ (visited on 02/14/2021).

[21] F. Ponce Mella, G. Márquez, and H. Astudillo, "Migrating from monolithic architecture to microservices: A rapid review," in *Proceedings of the 38th International Conference of the Chilean Computer Science Society*, Sep. 2019.

[22] F. Martin and L. James, *Microservices - a definition of this new architectural term*, Mar. 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html (visited on 02/14/2021).

[23] M. Amundsen and M. Mclarty, *Microservice Architecture: Aligning Principles, Practices, and Culture*. Sebastopol, CA: O'Reilly Media, Inc, USA, Aug. 2016, ISBN: 978-1-4919-5625-0.

[24] P. Drews, I. Schirmer, B. Horlach, and C. Tekaat, "Bimodal Enterprise Architecture Management - The Emergence of a New EAM Function for a BizDevOps-based fast IT," Oct. 2017. DOI: 10.1109/EDOCW.2017.18.

[25] M. E. Conway, "How do committees invent," *design organization criteria*, 1968.

[26] D. Faitelson, R. Heinrich, and S. Tyszberowicz, "Functional Decomposition for Software Architecture Evolution," in, Jul. 2018, pp. 377–400. DOI: 10.1007/978-3-319-94764-8_16.

[27] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a Smart City Internet of Things Platform with Microservice Architecture," in *2015 3rd International Conference on Future Internet of Things and Cloud*, Aug. 2015, pp. 25–30. DOI: 10.1109/FiCloud.2015.55.

[28] L. Sun, Y. Li, and R. A. Memon, "An open IoT framework based on microservices architecture," *China Communications*, vol. 14, no. 2, pp. 154–162, Feb. 2017, Conference Name: China Communications, ISSN: 1673-5447. DOI: 10.1109/CC.2017.7868163.

[29] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47 980–48 009, 2018. DOI: 10.1109/ACCESS.2018.2866491.

[30] S. Zhelev and A. Rozeva, "Using microservices and event driven architecture for big data stream processing," in *AIP Conference Proceedings*, vol. 2172, Nov. 2019, p. 090 010. DOI: 10.1063/1.5133587.

[31] K. Miao, J. Li, W. Hong, and M. Chen, "A Microservice-Based Big Data Analysis Platform for Online Educational Applications," *Scientific Programming*, Jun. 2020, ISSN: 1058-9244 Pages: e6929750 Publisher: Hindawi Volume: 2020. DOI: 10.1155/2020/6929750.

[32] D. Staegemann, M. Volk, N. Jamous, and K. Turoski, "Exploring the applicability of test driven development in the big data domain," in *Proceedings of the 2020 ACIS*, Dec. 2020.

[33] K. Lehmann and A. Freymann, "Demo Abstract: Smart Urban Services Platform a Flexible Solution for Smart Cities," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, Apr. 2018, pp. 306–307. DOI: 10.1109/IoTDI.2018.00052.

[34] R. Peinl, F. Holzschuher, and F. Pfitzer, "Docker Cluster Management for the Cloud - Survey Results and Own Solution," *Journal of Grid Computing*, vol. 14, no. 2, pp. 265–282, Jun. 2016, ISSN: 1572-9184. DOI: 10.1007/s10723-016-9366-y.

[35] J. Scott, *Using microservices to evolve beyond the data lake*. [Online]. Available: https://www.oreilly.com/content/using-microservices-to-evolve-beyond-the-data-lake (visited on 02/21/2021).

[36] M. Kiran, P. Murphy, I. Monga, J. Dugan, and S. S. Baveja, "Lambda architecture for cost-effective batch and speed big data processing," in *2015 IEEE International Conference on Big Data (Big Data)*, Oct. 2015, pp. 2785–2792. DOI: 10.1109/BigData.2015.7364082.

[37] T. Zschörnig, R. Wehlitz, and B. Franczyk, “A Personal Analytics Platform for the Internet of Things - Implementing Kappa Architecture with Microservice-based Stream Processing,” in *Proceedings of the 19th International Conference on Enterprise Information Systems*, Jan. 2017, pp. 733–738. DOI: 10.5220/0006355407330738.

[38] Z. Tejada, *Big data architectures - Azure Architecture Center*. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data (visited on 02/14/2021).

[39] A. Ali and M. Abdullah, “A Survey on Vertical and Horizontal Scaling Platforms for Big Data Analytics,” *International Journal of Integrated Engineering*, vol. 11, Sep. 2019. DOI: 10.30880/ijie.2019.11.06.015.

[40] S. J. Fowler, *Production-Ready Microservices: Building Standardized Systems Across an Engineering Organization*, 1st edition. Sebastopol, CA: O’Reilly Media, Dec. 2016, 4. Scalability and Performance - Production-Ready Microservices, ISBN: 9781491965979.

[41] D. Taibi, V. Lenarduzzi, and C. Pahl, “Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation,” *IEEE Cloud Computing*, vol. 4, no. 5, pp. 22–32, Sep. 2017, Conference Name: IEEE Cloud Computing, ISSN: 2325-6095. DOI: 10.1109/MCC.2017.4250931.