# Microservices for Redevelopment of Enterprise Information Systems and Business Processes Optimization

Robert Stricker[1], Daniel Müssig[2] and Jörg Lässig[1,2]
[1]Institutsteil Angewandte Systemtechnik AST, Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung IOSB, Am Vogelherd 50, Ilmenau, Germany
[2]University of Applied Sciences Zittau/Görlitz, Brückenstrasse 1, Görlitz, Germany

Keywords: Microservices, Cloud Computing, Legacy Systems, Enterprise Architectures, Business Processes, ITIL, Service Management.

Abstract: Due to cost pressure and static technological development, the lifecycle of large enterprise information systems in operation is coming to an end. At the same time and as part of possible solutions, the demands for cloud systems in the enterprise context is continuously growing. Although microservices have become an established architectural pattern used by well-known companies, many especially smaller corporations are shying away from using them. In this paper we present the positive and negative effects of converting legacy applications into cloud-based microservice architectures. In addition to technical aspects such as maintainability and scalability, organizational consequences are considered and analyzed. Furthermore, the positive effects on existing business processes, especially ITIL Service Management Processes, are addressed and it is demonstrated how ITIL metrics such as MTRS, MRTT or TRD can be optimized by using microservices. We show advantages of a microservice architecture in the optimization of existing business fields and how new business areas can be opened up easier compared to conventional enterprise architectures. Even if microservices are not a silver bullet, they should be considered and evaluated as an opportunity for a new software lifecycle of a legacy enterprise application or as an architectural pattern for profound redevelopment.

## 1 INTRODUCTION

Within the last few years, the microservice architecture has become established for a huge set of applications. Many well-known companies such as Netflix or Amazon base much of their success on the use of microservices. Several other corporations currently realize that their applications are approaching the end of the software lifetime (Ganesan and Chithralekha, 2016) and are no longer up to today's demands. Especially the inclusion of the cloud into existing legacy systems causes technical problems and in addition, a significant change in customer requirements can be seen.

It has almost become a matter of course that an application can be accessed from anywhere and the response time has to be very short, data must not be lost and the system must be available without interruption. Furthermore, the pay-per-use model continues to spread, since the customer is only willing to pay for the parts of an application that he actually uses, while the desire for customization continues to grow.

We are also recognizing a dramatic change in the demands of enterprises, especially in conservative sectors such as the utilities industry. IT systems are no longer only supporting regular processes, they are increasingly representing the actual value-adding processes in more and more industries. It also has to be reasonable to implement comprehensive analyses and evaluations as easily and automatically as feasible, and errors must be detected and eliminated as soon as possible.

We are currently observing all of this at the same time. This results in the desire to replace legacy systems with new, modern ones that meet the new requirements (internal and external). Also, this change involves a great deal of effort so that possible alternatives are carefully evaluated. We have determined that when assessing microservice architectures, often only the technical aspects are taken into account (Dragoni et al., 2017b), and usually, solely the positive features are highlighted (Hasselbring and Steinacker, 2017). In this paper, we want to present not only the challenges of introducing microservices, in particular for corporate organizational structures but also the many positive effects on business processes that may arise.

Stricker, R., Müssig, D. and Lässig, J.
Microservices for Redevelopment of Enterprise Information Systems and Business Processes Optimization.
DOI: 10.5220/0006791607190726
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)*, pages 719-726
ISBN: 978-989-758-298-1

We use a variety of ITIL processes as examples to show how they can be optimized and how related metrics can be improved. Primarily the latter offers high potential for optimization, as advantages can be achieved directly at the value-adding processes.

Modern architectures play an increasingly important role in the strategic orientation of enterprises. Barely any industry can afford not to deal with the current developments, which are supposed to be described with the slogans Cloud Computing, Big Data, Industrial IoT, Artificial Intelligence, or Smart anything. These requirements did not exist when the architectures of today's legacy applications were designed. It is therefore necessary to choose an architecture that can meet the new demands and responds as flexible as possible to changes in the future. We want to show that microservices can handle this not only at the technical but also at the process level and that a microservice architecture is worth serving as the basis for future concepts.

The remainder of this paper is organized as follows: Section 2 presents various projects and applications of microservices and the focal points of their work. The weaknesses of conventional enterprise architectures concerning technical characteristics and business processes are discussed in Section 3. In Section 4, we look at some critical points in the implementation of microservices within an organization and identify the difficulties that can arise. The resulting advantages, particularly for business processes, are presented in Section 5 as well as the opportunities to exploit new business areas. In Section 6 we conclude our main propositions and the aspects to be examined are shown.

## 2 RELATED WORK

There are several case studies, surveys, and approaches, which describe the stance of enterprises adopting to microservices or even accomplishing the transition from a monolithic application to microservices. Wittland and Steffens performed a survey among employees of several companies asking if they and their companies consider adopting microservices (Wittland and Steffens, 2015). Overall, microservices were rated very well. However, there were serious differences between employees and companies in the evaluation of the focal points. Companies aim to keep costs as low as possible, while employees rate flexibility, maintainability, and sustainability higher. The authors concluded from their studies that the optimal architecture has the following characteristics: low costs, high flexibility, good maintainability and easy migration. An architecture should be sustainable and easy to understand for employees.

Zúñiga-Prieto et al. describe in their successive papers a model-driven approach for the incremental integration of microservices into cloud applications (Zúñiga-Prieto et al., 2015) (Zúñiga-Prieto et al., 2016). They show that there is currently a lack of methodologies to integrate microservices into cloud environments. They proposed to model integrations first and then generate the necessary code.

Levcovitz and Valente proposed a technique to extract microservices from monoliths (Levcovitz et al., 2016). Facades, business functions, and database tables are modeled in a dependency graph in several steps in order to identify subsystems and later microservices based on them. They have tested their approach on a large distributed real monolithic banking system with about 750,000 lines of code. Among other things, they were able to identify 613 facades, 1131 business functions, and 198 database tables. Afterwards, they have successfully implemented several identified microservices.

Balalaie et al. have shared in their paper their experiences migrating to a cloud-native architecture (microservices architecture) (Balalaie et al., 2016b). They followed, among other things, the steps from Sam Newman's book (Newman, 2015) and showed the lessons learned. They noted that some additional services were needed, such as a service discovery and a load balancer. Also they remarked that compliance with the service contracts is very important and must not be underestimated, but also that developing in a distributed application presents special challenges to the developers.

Mazlami et al. proposed a tool to extract microservices from a monolithic application (Mazlami et al., 2017). They demonstrate a method which can be used to automatically extract microservices in a version control system. The method is based on the classes, the authors and the author's commits to the classes. They investigated three different extraction strategies, which they also combined among others. Then they applied these to 21 different open-source projects and examined the strategies on their performance according to custom metrics. They discovered a reduction of the team size to a quarter and reduce domain redundancy by 70 percent.

Hasselbring and Steinacker describe the usage of microservices at otto.de, one of the largest e-commerce platforms in Europe (Hasselbring and Steinacker, 2017). They demonstrated the successful reimplementation of a online shop system and the resulting positive changes through the use of microservices. Among other things, they were able to increase the number of weekly live deployments from 40 to about 500 with a tiny amount of live incidents. It was not only the improved scalability and agility from a technical point of view but also the better adaptability with the number of teams and developers in teams that persuaded them to switch to the new architecture.

## 3 WEAKNESSES OF CONVENTIONAL ENTERPRISE ARCHITECTURES

At first, we will focus on traditional architectures that are not or not yet running in the cloud.
Many negative aspects of an architecture results from the size of the artifacts, which can be considered as a unit due to their close coupling and dependencies as shown in Figure 1.

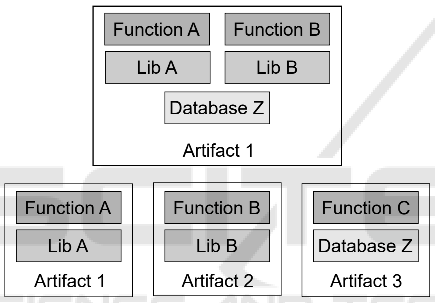

Figure 1: Most weak points in common architectures are based on the size of individual artifacts and the tight coupling of functions, whereas a microservice architecture results in smaller independent artifacts.

First of all, this size influences basic metrics such as startup or shutdown time which has a huge impact on other essential aspects of the life cycle of an application. Especially in the processes of service validation or testing, developers have to compile and execute the code again and again. Integration tests and system tests usually have to be carried out overnight because they take several hours. If errors occur, the release often has to be postponed by at least one day. In addition to the long execution time of the tests, the vast code base makes it difficult to find the causes of errors. This includes bugs that are not discovered until the application is deployed to the production system. In the case of large monolithic applications, significant changes usually lead to big-bang releases, which is not suitable for a modern change or release management according to a survey of Heusingfeld[1]. These entail many risks, as the effects of changes in one part often cannot be assessed against others. In addition, the boundaries of individual modules become blurred much more easily, which means that supposedly small changes can have a significant impact on other parts of the application.

[1]https://jaxenter.de/microservices-warum-57795

Another important point is the training of new employees, which is made unnecessarily difficult by large artifacts. The dependencies have to be mentally resolved, therefore a large number of libraries have to be considered. Heusingfeld also describes the unattractiveness for clients and applicants as another reason for turning away from conventional architectures.

Furthermore monolithic architectures are not technology-agnostic (Dragoni et al., 2017a) or software-stack-agnostic. The latter means that the application depends heavily on the software to third party components. If the provider of these external parts stops developing the software or prohibits the use of the software for any reason, the entire monolithic application has to be changed to another library within a very short period of time.

Turning to the cloud is a trend that has been going on for several years. Thus the companies want to save costs and at the same time achieve greater scalability and availability. However, the same legacy monolithic architectures are still used in many cases and are just copied to a virtualized environment (Balalaie et al., 2016b). The most significant issue with that is that a cloud infrastructure can fail at any time. Therefore, the application has to be designed to handle these circumstances. In addition, scalability can only be achieved if the architecture is also scalable. Thus the startup and shutdown times of an application should be as short as possible. Some cloud providers are now accounting on a per-minute basis, so the operational time of the application should be at an optimum not more than that. Every minute that is needed to start or stop the application, no consumers can be served and the company loses money during this time.

## 4 CHALLENGES WHILE ADOPTING MICROSERVICES

### 4.1 Transformation of Corporate Organization

Even though microservices are currently a technology-driven trend, technical changes are not the only ones approaching a company that wants to use a microservice enterprise architecture. A positive aspect is often suggested when the enterprise organization is adjusted (Hasselbring and Steinacker, 2017). We believe that this change is essential if microservices should be used permanently and with a maximum efficiency, as the corporate structure has a direct impact on the produced products, mentioned in Conways Law (Conway, 1968). The significantly shorter communication channels, as shown in Figure 2, have a positive impact on both product quality and business processes, which is discussed in detail in Section 5.2.

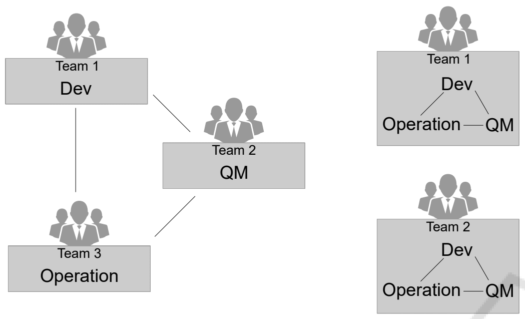

Figure 2: In order to fully exploit the advantages of microservices, it is necessary to transfer the corporate structure from a departmental organization (left in the figure) to a product-oriented organization (right in the figure).

Since different microservices can use completely different technology stacks, the product-oriented organization results in advantages from a competence point of view. Each team only has to deal with the technology needed for its microservice, while heterogeneity and complexity within a corporation is increasing.
The company is faced with a great deal of effort in carrying out this restructuring, especially when both legacy monoliths and novel microservice architectures are present in the portfolio. In addition, capable employees are needed, since either a certain distance from the specialization is taken (one product team consists only of software developers who perform all three DevOps (Balalaie et al., 2016a) tasks (development, quality management, operation) or specialists from their fields must work closely together, which requires a high degree of social competence. The teams should therefore be built up from experienced employees at the beginning (Taibi et al., 2017), while after the introductory phase new employees can be ideally trained. By working in different teams they can get knowledge about a lot of specialized technologies, agile software development, thoroughly validation and reliable operation of services. In this way, the qualification of employees can be significantly improved compared to relatively strict division into departments.

## 4.2 Entrusting the Cloud with Corporate Values

When we talk about legacy systems and optimizing business processes, we must not only look at start-ups and small, dynamic companies (e. g. in the software development industry). A large number of companies, primarily in very conservative sectors (energy supply, craft, production) are still afraid to store relevant company data in the cloud. In Germany in particular, this can be described as a factor hindering development and progress. This is often based on the mistaken assumption that the term ”cloud” is only used to describe the public cloud (Amazon AWS, Google or Azure). However, there are also other forms: private cloud (own infrastructure within the company) and hybrid cloud (a combination of both). Accordingly, sensitive data can be protected by a private cloud in which the microservice architecture can be implemented to achieve the advantages presented in Section 5 of this paper. This was implemented in one of our projects for an energy supplier and fully complies with the legal data protection requirements. Section 5.3 deals with the possibilities of new business areas through microservices, which can also be seen as advantages of using a cloud (regardless of which type of cloud is used).
In addition to privacy, security also plays a major role. Microservices generate more complexity (Balalaie et al., 2016b) due to the larger number of interfaces, which is reflected in a higher risk for security incidents. This has to be taken into account through comprehensive safety and risk management.

## 4.3 Eventual Consistency

Another important aspect that should be considered when introducing microservices is the so-called eventual consistency or adherence to the CAP-theorem (Brewer, 2000). This includes a somewhat depressing statement: a maximum of two of the three points consistency, availability and partition tolerance can be fulfilled completely. Consequently, if a legacy system gets redesigned, it must be borne in mind that not all three points are feasible, which is a question of the demands placed on the system. If the majority of operations are transactions, whereas availability is less important, microservices may not be the right way. However, it can be observed that most of the new business areas (see Section 5.3) are mainly committed to high availability and reliability.

## 5 ADVANTAGES OF A MICROSERVICE ENTERPRISE ARCHITECTURE

### 5.1 Technical Improvements

The greatest advantage of microservices compared to monoliths is the scalability (Dragoni et al., 2017b) (Müssig et al., 2017). Smaller systems are much more scalable because there are fewer dependencies on system environments and only the functions (services) that are in greater demand have to be scaled. This makes microservices ideal for products that are intended to appeal to a large number of customers or, as in the case of Smart Grid, have to deal with a very high number of measurement values. The resulting new opportunities for companies are examined in Section 5.3.

Another important technical advantage is the maintainability of an application. Although the complexity and effort of monitoring increases (Sun et al., 2015), errors can be assigned to a microservice more easily and, due to the small size, can usually be eliminated more quickly (Gouigoux and Tamzalit, 2017). In addition, services can be added or removed quickly and easily through containerization. Since microservices can be seen as the upper edge of the scale cube (Abbott and Fisher, 2015) and thus as the most extreme form of separation of functions (except serverless architectures (Baldini et al., 2017)), they are an extremely pronounced form of modularization, which also has a positive effect on maintainability. Testability also increases as a result of increasing modularization and is therefore very well integrated in microservices. Another important improved aspect concerns the quality attribute modernisability. In addition to the aforementioned features of microservices, complemented by the ability to use different programming languages, and the simple HTTP interfaces (when using RESTful microservices (Newman, 2015)), individual modules can be easily renewed and improved. New functions can be added step-by-step, which underlines the agile character also in the development process of the software (see Section 5.2). New services can also be operated in parallel to old versions, which is not possible with monoliths and can be used for new business areas, as described in Section 5.3.

### 5.2 Improvement of Business Processes

It is already known that many business processes can be optimized by adopting the cloud. Microservices are a cloud-native architecture (Balalaie et al., 2016b) so the circle is complete. But there are some other aspects that speak in favour of using microservices.

While in the first part of this paper we have focused on comparing microservice architectures with monoliths, we now want to show that existing service-oriented architectures can also be improved by using microservices. Therefor we are looking at ITIL IT service management processes and provide examples of how these processes can be optimized by using microservices.

**Service Design.** In Service Design, the processes service level management, capacity management and availability management can be improved due to the usage of microservices. The smaller services make it easier to differentiate between different quality and cost categories. Consider a service which is converting a given input to some output through different steps. By using Microservices and a pipes-and-filters pattern several service levels can be offered, which supports the increasing customer demand for a pay-per-use model.

Availability management is closely linked to service level management, since thresholds for availability are often defined via SLAs. Two important metrics can be optimized by using a microservice architecture. On the one hand, the number of service interruptions can be minimized, since microservices scale better compared to established architectural patterns (Taibi et al., 2017) and make it easier to create redundancies. On the other hand the mean time to restore service (MTRS) can also be significantly reduced, since microservices can be restarted very quickly (Dragoni et al., 2017a) due to the technology of containerization, without any negative impact on other services. This can also be done fully automatically and is used, for example, in the Docker Swarm[2].

The optimization of capacity management is closely linked to this issue. The really great scalability of microservices forms the basis for optimizing many processes like shown in Figure 3. Microservices can thus be scaled horizontally faster than other architectures (Hasselbring and Steinacker, 2017). This applies not only to the rapid start of a new service instance, but also to targeted monitoring and intelligent load balancing (Balalaie et al., 2016b), so that the number of incidents due to capacity shortages can be reduced. Capacity bottlenecks can not only be better predicted, but also quickly eliminated due to the fast start times of microservices and the low effort involved in integrating a new service instance, which reduces the Resolution Time of Capacity Shortage.

[2]https://docs.docker.com/engine/swarm/

**Service Transition.** The simplified and optimized deployment has already been considered from a technical point of view in Section 5.1. As a result, deployment management can also be improved and adapted to the increased customer requirements. Like shown in Figure 3, frequent releases are possible and allow for real a continuous improvement, not erratic updates (Taibi et al., 2017).

In addition, the total release downtime (TRD) can be reduced, since the improved deployment management and smaller service releases have a smaller scope - the effects are therefore limited and releases can be carried out more quickly. It is obvious that the risk of a system failure is much higher if changes are made in many places by means of rare, large releases compared to the adaptation of individual, small components.

Accordingly, change management is actively supported by microservices. Changes can not only be brought to market more quickly, which improves the mean request for change turnaround time (MRTT), they can also be developed faster as we show in Section 5.1. This is also achieved by the optimized company organization, which guarantees short communication channels and fast development cycles (see Figure 2).

The service transition also includes validation and testing of the services. Microservices are easier to test, especially in the area of component testing (Newman, 2015). For validation, e. g. of service levels, it can be added that microservices can be implemented relatively quick due to their simple interfaces and low functionality. The shorter development cycles result in a further advantage that is known from agile software development. In this way, results can be presented to the customer more quickly and the feedback can be returned to the developer more quickly, which includes many of the processes already discussed. However, it also has an impact on the metrics of the passed acceptance tests, as customer requirements can be implemented even more effectively. The improved testability due to the increased modularization has a positive effect on change management as the number of failed changes can be reduced. It becomes clear that the processes are closely connected and microservices provide benefits at critical places, enabling a variety of different processes to be optimized, which is illustrated in Figure 3.

**Service Operation.** The final point is service operation. The transformation of the corporate organization from a departmental organization to a product-oriented design (DevOps) presented in Section 4.1 results in many advantages. Not only are develop-

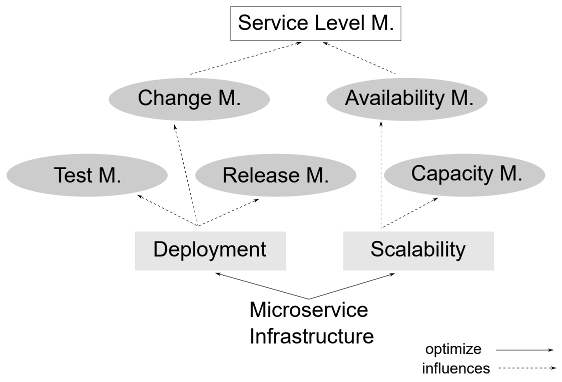

Figure 3: Due to their technical advantages (small functional scope, containerization) Microservices achieve an improvement of central capabilities such as scalability and easy deployment, which in turn influence many relevant processes.

ers encouraged to write higher quality source code to avoid being woken up by incidents at night (quotation M. Fowler), but entire process chains can be optimized. The communication channels of Incident Management, for example, can be shortened considerably (like shown in Figure 2), and any errors that may have occurred can be quickly eliminated by adding the processes mentioned above. It shows that many processes can be optimized through the use of microservices and thus offer customers added value, which has a direct positive effect on the company.

## 5.3 Developing new Business Areas

In the strategic orientation of companies, terms such as Industry 4.0, Industrial IoT, Smart followed by something (e. g. Smart City, Smart Grid, Smart Home) or artificial intelligence play an important role. What they all have in common is that the cloud is used for at least data aggregation and in most cases for much more complex business logic. While monolithic applications are less suitable for this, microservices are a cloud-native architecture (Balalaie et al., 2016b). Due to their size, independence through containerization, and communication via simple interfaces, they are also ideal for distributed systems. Modern applications consisting of automated data acquisition including pre-processing and detailed business logic in the cloud are therefore ideal for implementation. Examples of this are Smart Cities (Krylovskiy et al., 2015) or Smart Grids (Bottaccioli et al., 2017), but also energy management systems or applications in the field of ambient assisted living.

Compared to monoliths, functions can be designed, developed and, in most cases, released more independently, so the development phase of the product lifecycle can be significantly optimized and the time-to-market can be reduced.

In addition to the completely new areas of business that can be developed, there is also room for optimization potential in existing business fields. As described in the previous section, smaller services can be offered individually. Consequently, a large number of individual products are created from a former main product, which can be sold individually. Nevertheless, the compositions, i. e. products consisting of a variety of microservices, are also marketable. New compositions can be created easily by the simple interfaces and the loose coupling, see Figure 4. The result is a dynamic environment from which new products can be created.

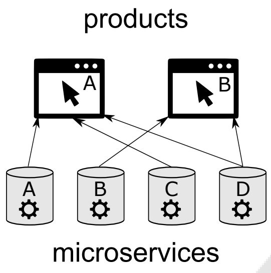

Figure 4: The simple interfaces and the loose coupling make it possible to create new products by combining existing microservices, which only have to be scaled horizontally if it is necessary.

In contrast to the use of a monolith, it is also possible to operate microservices in parallel in different versions. This has the advantage that newer services can also be marketed in a different way without forcing users to update, new service levels are thus created automatically by the improvement process of a service. The customer orientation can thus be clearly optimized and the analysis of demand is much more detailed than with monoliths.

## 5.4 Improvement of Security Features

It is often pointed out that a microservice architecture has weak points in the area of security (Dragoni et al., 2017a). We also mentioned this in Section 4.2, but the benefits of microservices in terms of security should not stay unmentioned. If one considers the three most important security goals confidentiality, integrity and availability (CIA-Triad), it can be determined that one central goal can be clearly optimized by using microservices. They make it easier to implement highly available systems than monoliths, as illustrated in Section 5.1 and Section 5.2. Additional effects can be achieved by applying the Security-by-Design principle when designing a microservice architecture (Müssig et al., 2017). Loose coupling between each other provides increased robustness, while in the case of a monolith, the failure of one component usually results in a complete failure of the system, microservices can be encapsulated in such a way that both failure and compromise of an individual service have no effect on other services. This is useful, for instance, if individual services are used to collect data and others to process data. Even if the data processing fails, the data is safely recorded. Nevertheless, more research is still needed in the field of microservice security to ensure that all security objectives are adequately met.

## 6 CONCLUSION

The microservice architecture was created in software development as a result of increasing modularization and a stronger focus on service orientation. Since microservice-based architectures become a serious alternative for common enterprise architectures and are evaluated as such, not only the technical aspects should be addressed.

Nevertheless, we have outlined the main advantages of microservices in this area because they represent the basis for process optimization as shown in Figure 3. In order to exploit these possibilities to their full potential, however, there are some hurdles to be overcome. In particular, the changes in the corporate organization from a departmental to a product-oriented structure and the associated overhead should be taken into account urgently before microservices are applied on a large scale within an enterprise. However, if this step is possible, there are many other advantages in addition to improved communication or real agile development which can be achieved.

Many business processes in the service lifecycle can be significantly improved, from service design to service operation. It can be seen that the technical advantages have a positive effect on some basic service management processes, which, due to their dependency on other processes, provides great potential for optimization, as described in detail in Section 5.2.

In addition, new business areas can be opened up, as microservices create a dynamic environment of small-scale functions in which new products can be built through combination. A library of products is formed, which can be managed by optimized processes and the customer orientation can be further improved.

Nevertheless, microservices are not a panacea - there will always be domains in which classical monoliths are preferable, for example if transaction safety is rated higher than availability. However, in addition to the technical aspects, the effects on business processes presented in this paper should also be taken into account when considering the introduction of a microservice-architecture.

## REFERENCES

Abbott, M. L. and Fisher, M. T. (2015). *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise*. Addison-Wesley Professional, 2nd edition.

Balalaie, A., Heydarnoori, A., and Jamshidi, P. (2016a). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software*, 33(3):42–52.

Balalaie, A., Heydarnoori, A., and Jamshidi, P. (2016b). *Migrating to Cloud-Native Architectures Using Microservices: An Experience Report*, pages 201–215. Springer International Publishing, Cham.

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., et al. (2017). Serverless computing: Current trends and open problems. *arXiv preprint arXiv:1706.03178*.

Bottaccioli, L., Estebsari, A., Pons, E., Bompard, E., Macii, E., Patti, E., and Acquaviva, A. (2017). A Flexible Distributed Infrastructure for Real-Time Co-Simulations in Smart Grids. *IEEE Transactions on Industrial Informatics*.

Brewer, E. A. (2000). Towards robust distributed systems. In *PODC*, volume 7.

Conway, M. E. (1968). How do committees invent. *Datamation*, 14(4):28–31.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., and Safina, L. (2017a). Microservices: yesterday, today, and tomorrow. In *Present and Ulterior Software Engineering*, pages 195–216. Springer.

Dragoni, N., Lanese, I., Larsen, S. T., Mazzara, M., Mustafin, R., and Safina, L. (2017b). Microservices: How To Make Your Application Scale.

Ganesan, A. S. and Chithralekha, T. (2016). A Survey on Survey of Migration of Legacy Systems. In *Proceedings of the International Conference on Informatics and Analytics - ICIA-16*, pages 1–10, New York, New York, USA. ACM Press.

Gouigoux, J.-P. and Tamzalit, D. (2017). From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture. *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, pages 62–65.

Hasselbring, W. and Steinacker, G. (2017). Microservice Architectures for Scalability, Agility and Reliability in E-Commerce. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, pages 243–246. IEEE.

Krylovskiy, A., Jahn, M., and Patti, E. (2015). Designing a Smart City Internet of Things Platform with Microservice Architecture. In *2015 3rd International Conference on Future Internet of Things and Cloud*, pages 25–30. IEEE.

Levcovitz, A., Terra, R., and Valente, M. T. (2016). Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems. *CoRR*.

Mazlami, G., Cito, J., and Leitner, P. (2017). Extraction of Microservices from Monolithic Software Architectures. *2017 IEEE International Conference on Web Services (ICWS)*, pages 524–531.

Müssig, D., Stricker, R., Lässig, J., and Heider, J. (2017). Highly scalable microservice-based enterprise architecture for smart ecosystems in hybrid cloud environments. In *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems*, volume 3.

Newman, S. (2015). *Building microservices: designing fine-grained systems*. O'Reilly Media, Inc.

Sun, Y., Nanda, S., and Jaeger, T. (2015). Security-as-a-Service for Microservices-Based Cloud Applications. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 50–57. IEEE.

Taibi, D., Lenarduzzi, V., Pahl, C., and Janes, A. (2017). Microservices in agile software development. In *Proceedings of the XP2017 Scientific Workshops on - XP '17*, pages 1–5, New York, New York, USA. ACM Press.

Wittland, J. and Steffens, A. (2015). Adoption of emerging Architectural Approaches in German Software Companies. *Full-scale Software Engineering*.

Zúñiga-Prieto, M., Abrahão, S. M., and Insfrán, E. (2015). An incremental and model driven approach for the dynamic reconfiguration of cloud application architectures. In *Information Systems Development: Transforming Healthcare through Information Systems (ISD2015 Proceedings)*.

Zúñiga-Prieto, M., Insfran, E., Abrahao, S., and Cano-Genoves, C. (2016). Incremental Integration of Microservices in Cloud Applications. In *Information Systems Development: Complexity in Information Systems Development (ISD2016 Proceedings)*.