Analyzing the Relevance of SOA Patterns for Microservice-Based Systems

Justus Bogner[1,2], Alfred Zimmermann[1], and Stefan Wagner[2]

[1] Reutlingen University of Applied Sciences, Germany
{justus.bogner,alfred.zimmermann}@reutlingen-university.de
[2] University of Stuttgart, Germany
{justus.bogner,stefan.wagner}@informatik.uni-stuttgart.de

Abstract. To bring a pattern-based perspective to the SOA vs. Microservices discussion, we qualitatively analyzed a total of 118 SOA patterns from 2 popular catalogs for their (partial) applicability to Microservices. Patterns had to hold up to 5 derived Microservices principles to be applicable. 74 patterns (63%) were categorized as fully applicable, 30 (25%) as partially applicable, and 14 (12%) as not applicable. Most frequently violated Microservices characteristics were *Decentralization* and *Single System*. The findings suggest that Microservices and SOA share a large set of architectural principles and solutions in the general space of Service-Based Systems while only having a small set of differences in specific areas.

Keywords: Microservices, SOA, Service-Based Systems, Design Patterns

1 Introduction

Over the last decade, Service-Oriented Computing (SOC) [13] established itself as one of the most important paradigms for distributed systems. The implementation of enterprise-wide software landscapes in the style of Service-Oriented Architecture (SOA) [3] brought benefits with respect to encapsulation, interoperability, composition, reuse, loose coupling, and maintainability. However, increased standardization and governance efforts, higher architectural and technological complexity, and sometimes vendor or product lock-in caused frustration and lead to failed SOA adoption projects [11]. In recent years, Microservices [7,12] as an agile, DevOps-focused, decentralized service-oriented variant with fine-grained services quickly gained in popularity and tried to address some of the issues with both large monolithic applications as well as "traditional" Service-Oriented Systems based on SOAP/WSDL and a central Enterprise Service Bus (ESB).

There is still an ongoing discussion in industry and academia about the differentiation of SOA and Microservices. Some see it as a very new architectural style that needs to be treated very differently ("revolutionary" perspective), some see it merely as a specialization of SOA, e.g. "fine-grained SOA" ("evolutionary" perspective). While many papers have been published on the subject, so far no comprehensive pattern-based approach to compare the two has been taken. Software design patterns are a well-established form to document proven solutions to recurring design problems within a specific context in a technology-agnostic yet easily implementable way. They base their origin in Alexander's building pattern language [1] and went mainstream with the famous Gang of Four "Design Patterns" [8]. There is large catalog of documented service-oriented patterns that emerged over the years as a result of growing SOA industry experience. However, it is not fully clear, if these patterns are of value for Microservice-Based Systems.

The contribution of this work is to add a pattern-based perspective to the "SOA vs. Microservices" discussion by analyzing the applicability of existing SOA patterns for a Microservices context. To create a basis for important principles of Microservice-Based Systems, Section 2 introduces existing comparisons of the two service-based architectural styles. Section 3 outlines the detailed scope and research method of the pattern-based approach, while Section 4 presents the results. Finally, Section 5 closes with a summary, limitations, and an outlook on potential follow-up research.

2 Related Work: SOA vs. Microservices

Several perspectives on the comparison of SOA and Microservices have been published so far. Zimmermann first compares the two most popular definitions of Microservices, namely the one of Lewis/Fowler and the definition of Newman [19]. He distills common tenets and warns that the two definitions mix concerns related to process, organization, architecture, and development, which should be avoided when defining an architectural style. He then analyzes the identified tenets for SOA pendants and finds similarities for most of them. For him, the largest differences show with respect to decentralized governance, infrastructure automation, independently deployable services, and lightweight communication as opposed to a central ESB. Based on these findings, the analysis concludes that Microservices can be seen as a specific development and deployment approach for SOA.

Similarly, Dragoni et al. come to the conclusion that Microservices are "the second iteration" of the SOC and SOA concepts with the aim to strip away complexity and to focus on the development of simple and lightweight services [2]. While SOA addresses the enterprise workflow level, Microservices aim for a smaller application-level scope. Other apparent differences for them include independent bounded contexts with small services, the high degree of automation, the organizational aspects related to DevOps teams ("you build it, you run it"), the preference of choreography over orchestration, and a potentially higher degree of technological heterogeneity.

Xiao et al. describe Microservices and SOA as allies that should be leveraged to enable a bi-modal or two-speed IT in the digital age [18]. Their comparison highlights autonomy, size, and the development and deployment cycle as main differences. Additionally, decentralized governance and different communication and message exchange protocols are pointed out. Apart from these, a lot of similarities are mentioned, especially the focus on organizing services around business capabilities and service-oriented principles like statelessness, reuse, and abstraction.

Salah et al. take a broad evolutionary perspective and compare the client/server architecture, mobile agents architecture, SOA, and Microservices [17]. They describe a direct line of evolution from client/server to SOA and from there on further down to Microservices. Main differences for them include high service independence and decentralization, fine-grained services with bounded contexts, fast software delivery, and lightweight communication via "dumb pipes" and no middleware focus.

Lastly, the most "revolutionary" perspective is taken by Richards [14]. He argues that SOA focuses on large, complex, enterprise-wide systems whereas Microservices target small to medium web-based applications. Likewise, SOA follows a "share-as-much-as-possible" approach while Microservices are based on the "share-as-little-as-possible" principle. For Richards, Microservices are located on the other side of the service-oriented spectrum as SOA. He focuses very much on differences, not so much on commonalities. Other notable Microservices differences presented are the low degree of centralization and standardization, a very small number of lightweight communication protocols as opposed to SOA's protocol-agnostic heterogeneous interoperability provided by an ESB, and the focus on bounded contexts as opposed to SOA's focus on abstraction and business functionality reuse.

To the best of our knowledge, there are currently no publications solely on the topic of SOA patterns and Microservices. There are some publications concerning patterns specifically tailored for Microservices [9, 10, 15], but the authors do not really explain the relation to SOA. Moreover, several of these Microservices patterns are not new and have been used in other contexts before (including SOA). A detailed analysis would be interesting, but goes beyond the scope of this paper.

3 Scope and Research Method

All comparisons in Section 2 focus on design characteristics, principles, or applied technologies. A different approach would be to analyze existing SOA patterns for applicability to a Microservices context. This architecture- and design-centric perspective has the positive side-effect of providing a list of candidate patterns potentially usable in a Microservice-Based System. We use Erl's [4, 5] and Rotem-Gal-Oz's [16] books as sources for SOA patterns, as they are well established in industry and academia and have minimal overlap. From literature (including the publications in Section 2) we compiled the following list of Microservice-specific principles.

- **Bounded Context:** fine-grained services according to Bounded Contexts [6]
- **Decentralization:** decentralization of control and management, low degree of standardization, choreography over orchestration

- **Lightweight Communication:** communication via RESTful APIs or lightweight messaging (no ESB, no workflow engine, etc.), "dumb pipes"
- **Single System:** building a single Service-Based System of medium size
- **Technological Heterogeneity:** support of diverse programming languages, databases, or used frameworks/libraries

With these criteria, we qualitatively analyzed the patterns in our 3 sources and documented if the usage of a pattern violates the compiled characteristics. Based on the violations, a pattern is categorized as **fully applicable**, **partially applicable** (with certain limitations/modifications), or **not applicable**. Some examples: The pattern *Enterprise Inventory* that provides architecture, standardization, and governance boundaries for every service within the enterprise is categorized as *not applicable*, because this violates the Microservice principles *Decentralization*, *Single System*, and *Technological Heterogeneity*. Likewise, the pattern *Protocol Bridging* that enables communication between consumers and providers that rely on different protocols is rated *partially applicable*, because this is usually not necessary in a Microservice-Based System. The principles *Lightweight Communication* and *Single System* could be violated by this. However, there could be rare evolutionary use cases where this pattern could indeed be applied, e.g. when a service that uses message-based communication (e.g. AMQP) should interact with one that relies on RESTful HTTP because of new or changed requirements. Lastly, the pattern *Lightweight Endpoint* where a series of fine-grained capabilities replaces a single coarse-grained capability to avoid wasteful data exchange and consumer-side processing is categorized as *fully applicable*. It violates none of the defined Microservices principles and is in fact in line with the core values of this architectural style.

The aggregated results of all these pattern categorizations are then used for further analysis and to provide answers to the following research questions:

RQ1: To what degree are SOA design patterns applicable to Microservices?

RQ2: What pattern categories are the most or least applicable?

RQ3: What are the most frequent Microservice-specific properties violated by not or partially applicable patterns?

4 Results: Applicability of SOA Patterns to Microservices

Erl's catalog comprises a total of 92 patterns (85 SOA patterns [4] and 7 REST-inspired patterns [5]). These patterns are structured into 5 different categories, namely *Service Inventory Design Patterns* (24 patterns), *Service Design Patterns* (31 patterns), *Service Composition Design Patterns* (23 patterns), *Compound Design Patterns* (7 patterns), and *REST-Inspired Patterns* (7 patterns). Rotem-Gal-Oz's more compact book presents 26 patterns [16] in 6 categories, namely *Foundation Structural Patterns* (5 patterns), *Performance, Scalability, and Availability Patterns* (6 patterns), *Security and Manageability Patterns* (5 patterns), *Message Exchange Patterns* (4 patterns), *Service Consumer Patterns* (3 patterns), and *Service Integration Patterns* (3 patterns). So all in all, we analyzed 118 SOA patterns with very few duplicates (examples being *Service Bus* or *Orchestration*).[3]

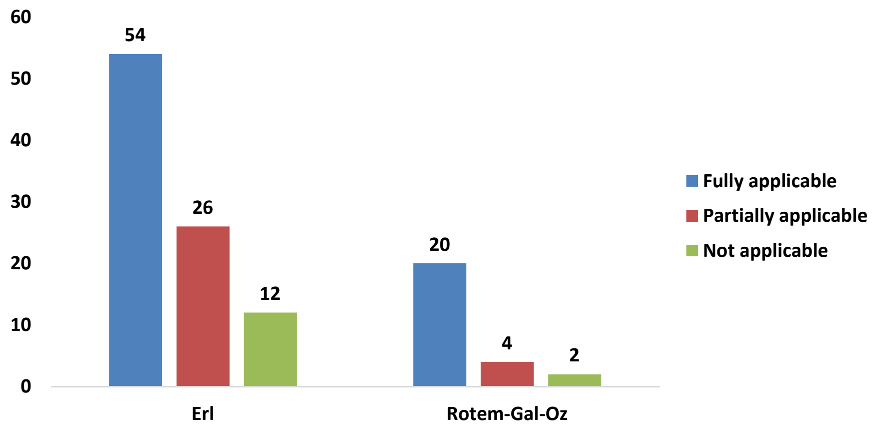

Fig. 1. SOA Pattern Applicability to Microservices

From Erl's 92 patterns, 54 (59%) patterns were found to be **fully applicable**, 26 (28%) to be **partially applicable** and only 12 (13%) were categorized as **not applicable**. That means that 87% of the patterns were estimated at least partially applicable in a Microservices context. For Rotem-Gal-Oz's smaller catalog, the numbers were even higher: Of the 26 patterns, 20 (77%) were categorized as **fully applicable**, 4 (15%) as **partially applicable**, and only 2 (8%) were deemed **not applicable**. So 92% of these patterns were found to be at least partially applicable for Microservices. When combining both catalogs (118 patterns), this accounts for 74 (63%) **fully applicable**, 30 (25%) **partially applicable**, and 14 (12%) **not applicable** patterns (see Fig. 1).

When looking at Erl's 5 pattern categories, an immediate observation is that all of the 7 *REST-Inspired Patterns* are fully applicable, which seems understandable in light of Microservice-related communication preferences. Furthermore, 22 of 31 *Service Design Patterns* (71%) and 15 of 23 *Service Composition Design Patterns* (65%) were fully applicable, which makes these two categories also very useful design sources when building Microservice-Based Systems. The 7 *Compound Design Patterns* were the least applicable category with 0 fully and only 4 partially applicable patterns (57%). This can be explained with the complexity and centralized nature of these patterns. For Rotem-Gal-Oz's 6 categories, both *Message Exchange Patterns* and *Service Consumer Patterns* are 100% fully applicable. Moreover, *Performance, Scalability, and Availability Patterns* with 83% and *Security and Manageability Patterns* with 80% fully applicable patterns

[3] For details see: https://github.com/xJREB/research-soa-patterns-for-microservices

are mentionable. All in all, the categories here consisted of fewer patterns, which makes it harder to compare them with Erl's.

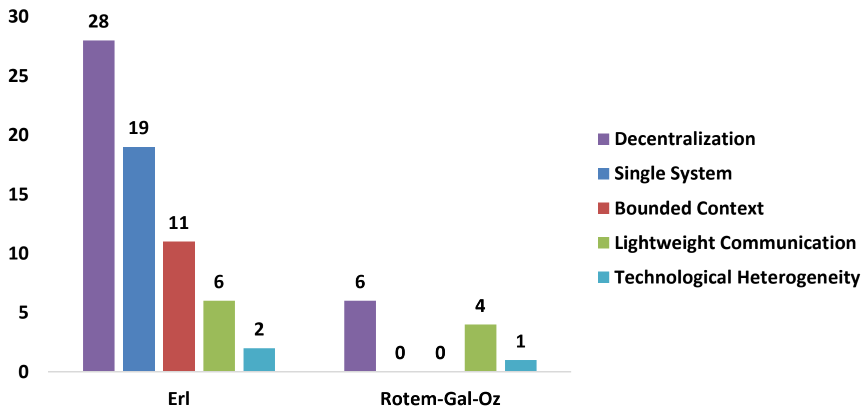

Fig. 2. Violated Microservice-related Principles of Not Fully Applicable SOA Patterns

When analyzing the most frequent causes why patterns were not or only partially applicable (see Fig. 2), the **Decentralization** characteristic was the most violated one (42% of all violations for Erl, 55% for Rotem-Gal-Oz). After that, **Single System** (29%) and **Bounded Context** (17%) were the next frequent violations in Erl's catalog, with **Lightweight Communication** only accounting for 9%. Interestingly, **Technological Heterogeneity** was violated only twice by Erl's patterns (*Enterprise Inventory* and *Domain Inventory*) and in Rotem-Gal-Oz's catalog only by one single pattern (*Service Host*). This can be explained with the technology-agnostic form of design patterns. Moreover, SOA systems are no strangers to diversity, so even an increase in technological heterogeneity in a Microservice-Based System will not invalidate the vast majority of patterns.

5 Summary and Conclusion

Based on 5 derived Microservices principles, we qualitatively analyzed the applicability of SOA design patterns for the context of Microservices. Of 118 patterns, 74 (63%) were found to be **fully applicable**, 30 (25%) **partially applicable**, and 14 (12%) **not applicable**. The most violated principles were *Decentralization* and *Single System* while *Technological Heterogeneity* had very little impact. The findings suggest that from a pattern-based perspective, Microservices and SOA have some small distinct areas of differences, but share a large set of design-related commonalities.

However, since descriptions of Microservices (unlike descriptions of “pure” architectural styles) cover other areas than architecture and design (e.g. process, organization, development, operations, etc.), differences in these other areas may be much more apparent. Since a definition of SOA as an architectural style should not define or restrict these areas, this seems to support the view of [19] that Microservices can be seen as a specific development and deployment approach for SOA.

Limitations of our work are the qualitative nature of the comparison. Results precision could have benefited greatly from a more rigorous method to rate patterns and state that a principle was violated. Similarly, an external validation of the pattern ratings by experts in the field of Microservices would have improved the results further. This would have reduced the possibility of subjective bias and increased the reproducibility of the study. Follow-up research could include such methods as well as trying to identify SOA patterns in existing Microservice-Based Systems. Lastly, it will be interesting to analyze the currently forming catalog of Microservices patterns to check for either “SOA backwards compatibility” or existing SOA pattern ancestors.

Acknowledgments This research was partially funded by the Ministry of Science of Baden-Württemberg, Germany, for the Doctoral Program “Services Computing” (http://www.services-computing.de/?lang=en).

References

1. Alexander, C., Ishikawa, S., Silverstein, M., i Ramió, J.R., Jacobson, M., Fiksdahl-King, I.: A Pattern Language. Gustavo Gili (1977)
2. Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R., Safina, L.: Microservices: Yesterday, Today, and Tomorrow. In: Present and Ulterior Software Engineering, pp. 195–216. Springer International Publishing, Cham (2017)
3. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River, NJ, USA (2005)
4. Erl, T.: SOA Design Patterns. Pearson Education, Boston, MA, USA (2009)
5. Erl, T., Carlyle, B., Pautasso, C., Balasubramanian, R.: SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST. The Prentice Hall Service Technology Series from Thomas Erl, Pearson Education (2012)
6. Evans, E.: Domain-driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley (2004)
7. Fowler, M.: Microservices Resource Guide (2015), http://martinfowler.com/microservices
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston, MA, USA (1994)
9. Gupta, A.: Microservice Design Patterns (2015), http://blog.arungupta.me/microservice-design-patterns
10. Krause, L.: Microservices: Patterns and Applications. Lucas Krause (2015)
11. MacLennan, E., Van Belle, J.P.: Factors affecting the organizational adoption of service-oriented architecture (SOA). Information Systems and e-Business Management 12(1), 71–100 (2014)

12. Newman, S.: Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, 1st edn. (2015)
13. Papazoglou, M.P.: Service-oriented computing: concepts, characteristics and directions. In: Proceedings of the 7th International Conference on Properties and Applications of Dielectric Materials (Cat. No.03CH37417). pp. 3–12. IEEE Comput. Soc (2003)
14. Richards, M.: Microservices vs. Service-Oriented Architecture. O'Reilly Media, Sebastopol, CA (2016)
15. Richardson, C.: Microservices Patterns. Manning Publications (2018)
16. Rotem-Gal-Oz, A.: SOA Patterns. Manning, Shelter Island, NY (2012)
17. Salah, T., Jamal Zemerly, M., Chan Yeob Yeun, Al-Qutayri, M., Al-Hammadi, Y.: The evolution of distributed systems towards microservices architecture. In: 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST). pp. 318–325. IEEE (2016)
18. Xiao, Z., Wijegunaratne, I., Qiang, X.: Reflections on SOA and Microservices. In: 2016 4th International Conference on Enterprise Systems (ES). pp. 60–67. IEEE (2016)
19. Zimmermann, O.: Microservices tenets. Computer Science - Research and Development 32(3-4), 301–310 (2017)