The Fault in Our Services: Investigating Integration-Relevant Faults in Microservices

Lena Gregor[1] · Alexander Pretschner[1,] · Anja Hentschel[2] · Horst Sauer[2] · Matthias Saft[2]

[1] Technical University of Munich, Munich, Germany
[2] Siemens AG, Munich, Germany

✉ lena.gregor@tum.de

Abstract. Microservices architecture has revolutionized software development, offering scalability, agility, and maintainability. However, integrating multiple microservices presents challenges, with faults impacting system reliability and performance. Testing the integration is vital to ensure seamless functioning, but knowledge about integration faults in microservice-based systems is limited. Existing taxonomies focus on non-functional attributes and lack structure for efficient test strategy development. In this extended abstract, we present our preliminary work to address this gap. Our approach involves adapting a Service-Oriented Architecture (SOA) fault taxonomy to the microservices context. By creating a structured overview of faults relevant to service integration, our aim is to empower practitioners and researchers in developing efficient testing strategies and assessments.

1 Introduction

Microservices architecture has transformed software development, offering scalability, agility, and maintainability. However, integrating multiple microservices poses challenges, with faults impacting the system's reliability and performance. Testing their integration is vital to ensure seamless functioning but their distributed nature, API interactions, and data consistency management introduce complexities. Faults arise from compatibility issues, network failures, data transformation, and service dependencies, leading to disruptions, failures, and degraded user experience.

Effective integration testing methods could help to identify, rectify, and prevent integration faults. Knowledge about the faults that can occur during the integration of services with other services or their environment can help to develop efficient test strategies in the development process. However, the knowledge about faults in microservice-based systems is sparse - especially with a focus on integration faults. To the best of our knowledge, there is only one work about the faults in microservice-based systems by Silva et al. [3]. This taxonomy focuses on the impact of faults on non-functional attributes like maintainability or performance. As the faults are not structured and often provide a different level of granularity or generalizability, this taxonomy does not facilitate a structured overview to create test strategies. Therefore, we want to create such an overview of faults that are relevant for microservice integration, which helps practitioners and researchers to enrich their knowledge about faults and create efficient strategies for testing and test assessment.

© 2023 The Authors. This is document is shared under the terms of the Creative Commons Attribution License (CC-BY 4.0), which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

In this extended abstract, we describe our preliminary work on the adaptation of an existing fault taxonomy for Service-Oriented Architecture (SOA) systems to the microservice context. In the first step, we adapted a SOA fault taxonomy [2] from the literature to the context of microservice-based systems with our knowledge from industry and academia. We provide our preliminary results of this step in this work, to open the topic for discussions within the community. In future work, we want to enrich the taxonomy with knowledge and insights of faults from real-world projects through interviews with practitioners. In the last step, we plan to validate the resulting taxonomy through a survey with practitioners and researchers.

In the following, we describe the adaptation of the original fault taxonomy for the SOA context to the microservice context.

2 Adapting the Original Taxonomy

We adapted a Service-Oriented Architecture (SOA) taxonomy that was published by Bruning et al. [2] in 2007. While using this rather old taxonomy may raise concerns about its alignment with the present-day landscape of SOA, it was a deliberate choice. During our literature search we did find a more recent publication of a SOA fault taxonomy by Bhandari and Gupta [1] from 2018. However, their work does not provide substantial advancements or insights about faults in current SOA systems – instead, they also used the taxonomy of Bruning et al. and added faults relevant to distributed systems in general and non-functional aspects to the existing taxonomy. Therefore, we opted to work with the original taxonomy, acknowledging its potential shortcomings. As we do not plan to focus our work on the differences between SOA and microservice systems but solely use this taxonomy as a starting point, due to the lack of equivalent taxonomies for microservice-based systems, we do not think that this is a crucial limitation of our work.

In the following, we describe the different fault classes in Bruning et al.'s taxonomy [2] and elaborate on their relevance for the microservice system context. The resulting first version of a taxonomy for integration-relevant faults for microservice-based systems is shown in Figure 2.

2.1 Publishing Fault

Bruning et al. describe in their work that during the publishing phase, a service is deployed on a server and its description is made public. According to them, the services have to be self-descriptive. Therefore, their description should include all syntactic as well as semantic information about the service. [2]

In practice, microservices can be openly published to be used by anyone – however, often times a microservice is also only used specifically in one system. If the term "publishing" is used in a microservice-based context, it is often used interchangeably for the term "deploying", which does not go in line with the meaning of the term "publishing" Bruning et al. describe. Furthermore, the sub-category *Service Deployment Fault* within the *Publishing Fault* category suggests, that the authors intended a difference between service publishing and service deployment in the SOA context. Although the top-level category *Publishing Fault* does not exactly fit the microservice context, the faults *Service Description Fault* and *Service Deployment Fault* and their "sub-faults" seem relevant, as we also have service descriptions and service deployment in this context. Therefore, we delete the top-layer category *Publishing Fault* and add the *Service Description Fault* and *Service Deployment Fault* as top-layer faults into the adapted taxonomy, also including their respective "sub-faults".

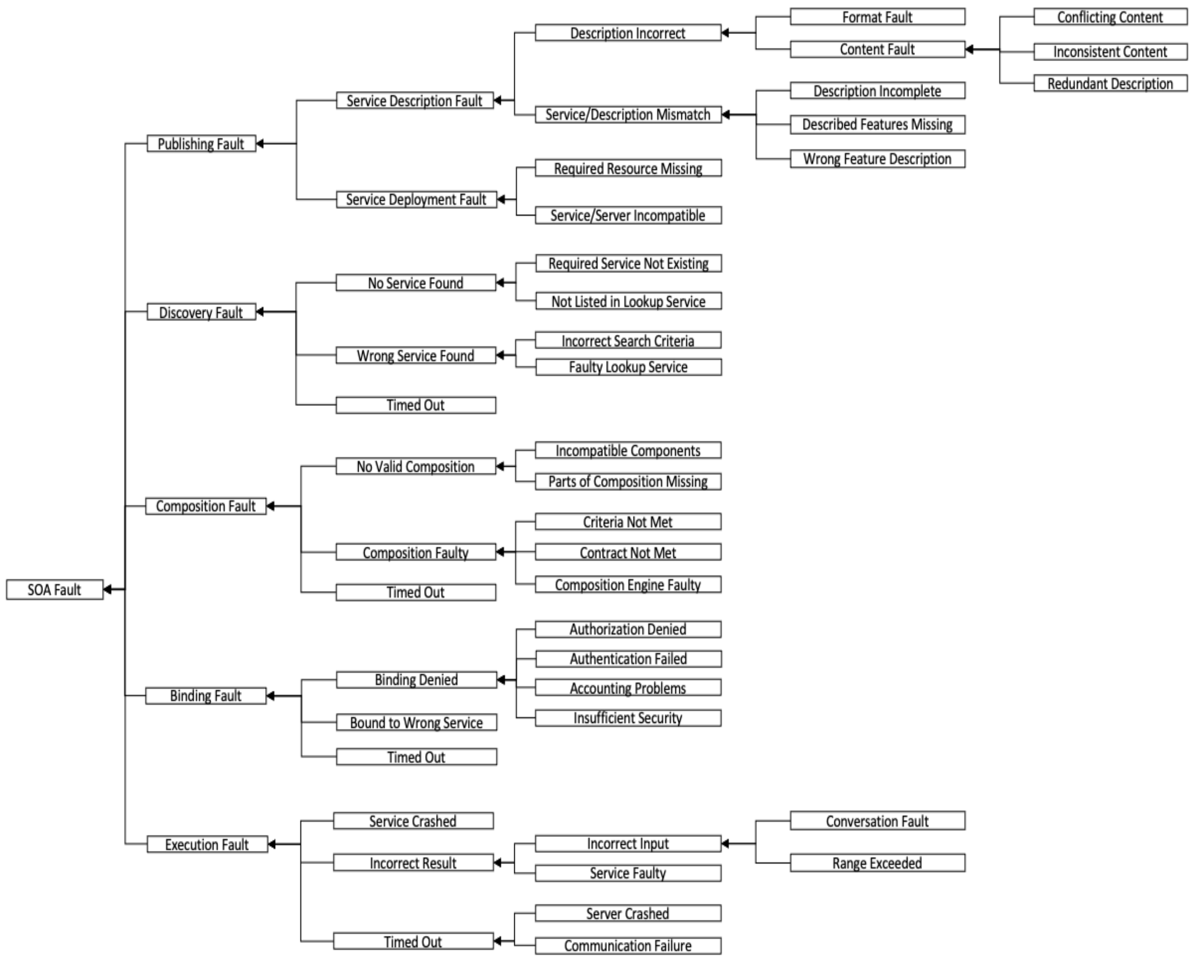

Figure 1: Original fault taxonomy for SOA systems by Bruning et al. [2].

As for the SOA context Bruning et al. assume that a provider is implemented independently from (potential) consumers, the original *Service/Description Mismatch* faults are focused around the viewpoint that the implementation of the service is correct and only the description can be incorrect [2]. However, in a context, where a microservice system is developed within a company, we can also have discussions and agreements between the development teams of two or more microservices, regarding which features and interfaces the services should offer. In this case, we see the feature description of a service as correct, but the implementation of the features as incorrect. Therefore, we add the *Wrong Feature Implementation* fault to the *Service/Description Mismatch* faults.

2.2 Discovery Fault

In the SOA discovery phase, a consumer looks dynamically for another service that provides the necessary functionality [2]. To the best of our knowledge, this is not yet feasible in practice. Instead, developer(s) choose the service that they want to use (either in-house or third-party). However, microservices often do not have a static address, in order to ensure flexibility and scalability. Instead, service registry components are used so that a service can look up the location (IP address and port) of another service or service instances. In this case, we can also have the faults *No Service Found* and *Wrong Service Found*, which is why we keep these fault classes, although the *Discovery* phase in a microservice context is slightly different than in a SOA context. As we do not dynamically search for a service that provides our needed functionality based on search criteria, the *Incorrect Search Criteria* fault is not relevant for the microservice context. However, through discovery, we could also get the wrong version of a service. Therefore, we add this fault to the *Wrong Service Found* fault class.

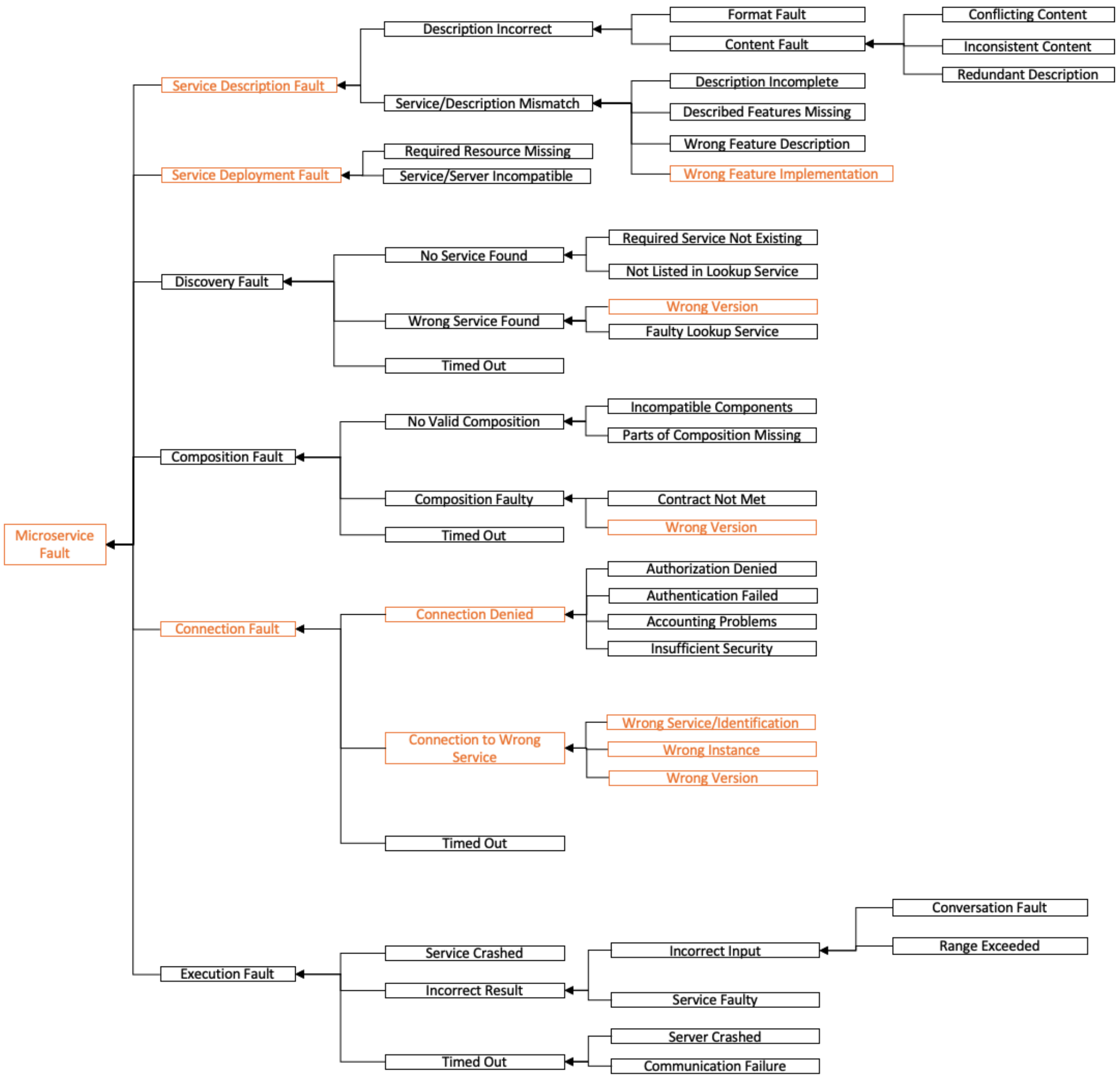

Figure 2: Adapted fault taxonomy for microservice-based systems. Changed or added faults are marked in orange.

2.3 Composition Fault

Again, for their fault taxonomy, Bruning et al. assume that in a SOA system, composition works dynamically – although they also state in their paper that this is yet not implemented in practice [2]. To the best of our knowledge, a dynamic finding and composing of multiple microservices to a combined functionality is not yet feasible in practice. However, microservices normally have dependencies on other microservices and provide a combined functionality, which is why we consider composition faults still relevant for the microservice context. However, without dynamic composition, we do not have a composition engine, which also means that we can not have a *composition engine fault*. Furthermore, as we do not use search criteria for the discovery, we also can not have a *Criteria Not Met* fault in the composition phase.

2.4 Binding Fault

In SOA, during the binding phase, the consumer binds itself to the service provider for the execution [2]. In the microservice context, we can have such a binding when we use e.g. gRPC or message queues as APIs. However, if we have a REST API, the term "binding" is not fitting. Therefore, we rename this phase to "Connection" which results in the fault class *Connection Fault* and its sub-classes *Connection Denied* and *Connection to Wrong Service*. In the microservice context, we can think of multiple sub-faults for the *Connection to Wrong Service* fault: (1) We can connect ourselves to the wrong service, if we have name duplicates, (2) we can connect ourselves to the wrong service instance, if our microservices are erroneously not stateless, (3) or we can connect to the wrong service version. Therefore, we add these faults to our fault taxonomy.

2.5 Execution Fault

During the execution phase, the service input parameters are transmitted from the consumer to the provider, the provider is executed and the output parameters are then returned to the consumer [2]. This phase can be directly transferred to the microservice context.

3 Conclusions

We have presented our preliminary work on the adaptation of a fault taxonomy from Service-Oriented Architecture to the microservice domain.This preliminary work is neither complete not empirically proved – however, we think it is a valuable first step to open discussions within the community. With future work, we plan to enrich the adapted taxonomy through empirical research and interviews with industry practitioners, to gain a deeper understanding of the faults that can occur during the integration of multiple microservices. Furthermore, we plan to validate the results through a survey with practitioners and other researchers from this community.

The resulting fault taxonomy can serve as a comprehensive framework for categorizing and classifying integration faults, enabling practitioners and researchers to identify, analyze, and address these faults more effectively. It provides a foundation for developing testing strategies, fault detection mechanisms, and fault tolerance techniques in the realm of microservices architecture.

References

[1] Guru Prasad Bhandari and Ratneshwer Gupta. Extended Fault Taxonomy of SOA-Based Systems. *CIT. Journal of Computing and Information Technology*, 25(4):237–257, January 2018. Number: 4.

[2] Stefan Bruning, Stephan Weissleder, and Miroslaw Malek. A Fault Taxonomy for Service-Oriented Architecture. In *10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*, pages 367–368, November 2007.

[3] Francisco Silva, Valéria Lelli, Ismayle Santos, and Rossana Andrade. Towards a Fault Taxonomy for Microservices-Based Applications. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*, SBES '22, pages 247–256, New York, NY, USA, October 2022. Association for Computing Machinery.