What Microservice Teams Need to Know of Each Other

Jonas Sorgalla[1]

[1] IDiAL Institute, Fachhochschule Dortmund, Dortmund, Germany

✉ jonas.sorgalla@fh-dortmund.de

Abstract. Building microservice-based applications inherently becomes an effort of multiple teams collaborating to achieve a common goal. While previous research has shown that the necessary alignment of organization and software architecture is a major challenge in the application of microservices, the act of collaboration has not been researched in detail. Therefore, we conduct a first qualitative study comprising five experts on which types of information should be shared between teams for successful microservice development. Our results show that the planned evolution of a microservice and the affiliation to business processes are of primary concern when it comes to information that should be shared between teams. Our findings can be used in tool development, to cross-reference information sharing behavior in practice, as well as foundation for future quantitative research.

1 Introduction

The microservice architecture style (MSA) has become the standard for complex distributed applications in practice. In particular, due to the scalability and availability opportunities opened up by MSA-based systems, the architecture style is the de facto standard for larger web applications such as e-Commerce or streaming platforms [8].

However, the quality features of a microservices system do not come without cost in the development process. The fact that microservices must be planned, developed, operated, and maintained as autonomous software modules increases the complexity of the task for those responsible in that a variety of skills and knowledge from all areas of the software lifecycle are required. A particular challenge here is that MSA also demands adaptation of the organization [5]. Indeed, in order to create truly autonomous software building blocks in the mid- and long-term, as well as to keep the coupling between microservices loose, this autonomy and loose coupling must also be reflected in the communication structure of the developing organization. This natural relationship, known as Conway's Law [3], ensures that for MSA development a structure of multiple teams emerges, where each team is fully responsible for one or more services, i.e., team members having all the competencies to support the microservice throughout its software lifecycle [8]. However, the structuring into different teams raises the question of how it is possible to align the teams towards a common goal while ensuring as much autonomy as possible so that the teams actually produce microservices that are as autonomous as possible. In this research, we contribute to answering this question by investigating what information should be shared between teams in a microservice development process, i.e., what should the teams know of each other? We do this by conducting a qualitative study in the form of expert interviews.

The remainder of this paper is structured as follows. In Section 2 we provide a brief overview of related empirical studies. Section 3 introduces the study's research design and Section 4 describes its execution. We present our results in Section 5. Finally, we discuss and conclude our findings in Section 6.

2 Related Work

Although we are not aware of any study in which information types are investigated between teams in the area of MSA development specifically, there is research in similar areas that relates to our study from the domains of large-scale agile development research [4], which generally deals with the application of agile principles in larger projects, as well as empirical research on MSA.

Regarding large-scale agile development, Dikert et al. conducted a systematic literature review on challenges and success factors for applying agile principles in multi-team scenarios [4]. As microservice teams are generally cross functional and apply agile principles [8], the results from Dikert et al. are relevant. The study shows that coordination challenges in a multi-team environment are quite common and were reported in 31% of the 42 studies investigated. Besides challenges such as the lack of standardized build scripts, synchronizing the definitions of software interfaces, and balancing team goals with organization goals, the study does not provide more details on the actual information that should be shared to mitigate named challenges. Moe et al. provide a study specifically on team autonomy in large-scale agile scenarios [7], which is of particular interest since MSA teams have to be autonomous by design. Their findings indicate the importance of sharing goals between such teams. They also find that teams need to communicate about access to shared resources or unclear requirements. As both studies indicate, interfacing between teams is not a challenge exclusive to organizations applying MSA but to large-scale agile development, in general. However, they naturally lack details on information types that are shared, such as those dependent not only on the organization but also the software architecture style of the system-under-development.

Concerning empirical research in the domain of microservices, there are also several related contributions. For example, Fritzsch et al. have conducted 16 interviews with practitioners regarding challenges, intentions, and strategies for microservice migration [5]. They identify organizational structure and team collaboration as crucial points for successful migration. Team communication as a research direction for MSA has already been identified by Lenarduzzi & Sievi-Korte [6]. They propose a large-scale multi-case study. The results of our study can be understood as a first step in this direction and serve as means to build proper instrumentation for a larger study.

3 Methodology

As our study aims to understand the nature and reason of communication between microservice teams, we leverage expert interviews, which is a survey method and can be categorized as a qualitative research approach which is particularly suitable for such exploratory purposes [10]. Using the GQM template for goal definition [1], the goal of our study can be summarized as follows.

Analyze *team communication in microservice development processes* for the purpose of *understanding team collaboration* with respect to their *shared information types* from the point of view of *process experts* in the context of *large-scale agile development of MSA*.

For data collection, i.e., conducting the interviews, we prepared a semi-structured guideline with the main focus on the what, why and how of information sharing between teams: (i) What information are important for teams to successfully collaborate for an MSA application? (ii) Why are they important? (iii) How should they be communicated; e.g., how should documentation look like?

Our guideline also includes an initial statement regarding work and MSA experience. Potential interview experts should at least have 3 years of experience working in organizations applying MSA as well as having held a position with process responsibility, i.e., they should have interacted with multiple teams and, thus, be knowledgeable of communication happening between teams. We recruited the interviewees in a non-probabilistic way based on existing contacts. Initially, we discussed MSA in general to ensure that interviewees shared a common understanding of the architectural style.

4 Execution

In total, we conducted five interviews in winter of 2022/2023 each lasting approximately one hour. Each interview was conducted via video call and recorded. When conducting the interview, we loosely followed our interview guideline and, depending on the type of information mentioned, asked clarifying follow-up questions. For analysis, we transcribed each interview and applied an inductive coding procedure comparing the types of information mentioned and the reasoning of each expert. We made the main questions of the survey known to the interview partners a few days before the interview so that they could organize and prepare their thoughts. Table 1 shows the demographic data of our interviewees. We labeled the interviewees from I1 to I5.

Id	Professional Background	Work Experience in years	Microservice Experience in years
I1	Tech Lead	10	7
I2	Full Stack Engineer	9	7
I3	Senior Architect	25	7
I4	Senior Architect	20	7
I5	DevOps Engineer	8	3

Table 1: Demographic data of interviewees.

5 Results

Table 2 lists all types of information that our experts have mentioned as relevant for microservice collaboration. In the following, we elaborate about the categories that we established to make the information types more manageable.

In general, our interviewees report that large scheduled and recurring meetings bringing together multiple members of all teams would not be purposeful and, in fact, counterproductive to team autonomy. The resulting personnel costs and the time to invest would probably even hinder effective development. More purposeful are planned and recurring meetings organized by function or expertise. I2 states in this regard, “the scrum masters from each team sit down together or the product owners from each team and then discuss something like the sprint focus, e.g., we want to make the

Category	Information Type	I1	I2	I3	I4	I5
Fundamental	Link to Codebase	•	•	•	•	•
	Link to Running Test Instance	•	•	•	•	•
	Technical Interface Description	•	•	•	•	•
Guidance	Guidelines for Interface		•			•
	Security Concerns	•				
Evolution	Planned Features	•	•	•	•	
	Single Point of Contact		•	•	•	•
	Purpose of Service	•		•	•	
	Quality Metrics	•				
Process Involvement	Consumed and Consuming Services	•	•	•	•	•
	Associated Business Processes	•		•	•	
Innards	Infrastructure-as-Code	•	•			•
	Technology Stack	•				•
	Design Decisions					•
	Risks					•
	Technical Debt					•
Runtime Behavior	Known Errors/Issues	•			•	
	Expected Operational Behavior				•	

Table 2: Identified information types for successful team collaboration in the context of MSA.

search better in the system." The people who attend such scheduled meetings then have a multiplier role within their respective teams. However, our experts agree that there should be a set of information that other teams can access on demand. This includes the majority of the information types mentioned to us.

[Fundamental] Among the information to be made available to other teams, our experts unanimously name *link to the code base* of the microservice in question, a *link to a running test instance* of the service, and a *technical interface description*, which should usually be available in the form of a self-describing interface accessible through the running instance, e.g., using OpenAPI[1], as fundamental for any team collaboration. These information enable other teams to be sure that they are actually implementing against the current interface and not using outdated documentation. It also enables early testing of the interaction of services.

[Guidance] In addition to the fundamental information about a microservice, I1 and I5 also describe guidance, such as a guideline on how to consume the service interface or which security concerns should be addressed when using the service, for other teams as helpful.

[Service Evolution] Our experts state that it is important that teams are able to inform themselves about the evolution of other teams' microservices in the future in order to take changes of microservices into account for their own development effort, e.g., when a microservice that the team's microservice depends on is changing the interface structure. In particular, most of our experts mentioned *planned features* and a general *purpose statement* of the service to be relevant for teams to assess the impact of evolution on their microservices. I1 mentions that, especially in larger systems, the purpose of a microservice is not enough and teams need to be able to gather information on *quality metrics* of the service, that is, what defines for the owning team whether the microservice im-

[1] `https://www.openapis.org/`

plementation is good or not, such as the service is optimized for quick loading time. Furthermore, all interviewees except I1 also mention the need for a *single point of contact*, i.e., a single person to contact when questions regarding a particular microservice arise, e.g., to clarify how the interface will evolve or when an expected breakdown occurs during operation.

[Process Involvement] All of our respondents said that it was important for teams to know which services their microservice depends on and which other teams use it in turn. This should be information that employees are actively informed about. I3 clarifies that this is important because it aligns a team with the common goal of the organization and, especially when services evolve, "I think I also have to know the dependencies to all calls, in order to know on whom I have an impact on." I1 mentions in this regard "you have to understand who is upstream and downstream [of your microservice]. However, that is a question that is not always trivial because sometimes, especially in larger organizations, you do not know anything about the teams with which your written software might interact with. But for business interest in general and also for the stack of your own team's standing in the organization, it can be quite important to investigate." I1, I3, and I4 also state that it is not enough to know the call chain but that it is also important for the creation of a common goal what place a team has in business processes. Here, I1 specifically suggests that teams should be shown their place in the *service blueprints* [2] of the product they develop for. I4 also mentions the technical necessity: "when we have more complex processes that go across different data stores or services, then it's very important to understand the process dependency because it's often necessary to do steps in a certain order because otherwise you kind of don't have a consistent process or a consistent data store anymore."

[Innards] Our experts attribute only a minor role to the inner structure of a microservice for collaboration with other teams. As I3 describes, "the relevant thing for outsiders is not how the microservice works inside, i.e., the code, how it is implemented, and so on. [...] Actually, I want to know what input parameters I have. What are the valid return parameters? How is the data structure?" However, as I1, I2, and I5 mention, it might be relevant to share information of the infrastructure that is necessary to run the service, best as *Infrastructure-as-Code*, e.g., Helm charts[2], when other teams want to run a foreign service for testing. I1 and I5 mention the technology stack as useful information because it enables teams to better look for best practices applied by others. Furthermore, I5 mentions attributes such as *design decisions*, *risks* and *technical debt*, however, he is not sure about their usefulness.

[Runtime Behavior] I4 suggests that "what else is interesting are questions regarding operational behavior, like how is the expected performance or what is the usual latency. If I have somehow detected a poor response time for my service, is it caused on my site or is it due to other microservices that I rely on?" In this sense, I1 and I4 both consider the information on known errors as important.

6 Conclusion

Our study identified 18 types of information that experts consider important to share between teams. Most crucial seem information that provide means for teams to work with another team's service in general. Although we expected process involvement to be named in the same capacity, we were surprised at how vehemently all experts made this point. Developing a shared vision is essential for autonomous teams to work efficiently in multi-team scenarios [4]. As we noted in a previous study, smaller organizations that develop MSA in particular seem to be challenged by this tension between alignment and autonomy [9]. Making transparent the process to which the team's microservice con-

[2] `https://helm.sh/`

tributes in the big picture seems to be an aid addressing this issue. The third important category we identify is information about the future evolution of other services because it may influence the future evolution of a team's microservices due to service dependencies.

Regarding the validity of our study, we rate threats to internal validity as low as we recorded and transcribed the interviews, gave our participants the chance to feedback the transcribe, and our study's general more descriptive focus. However, there might be a participant bias in which experts have not mentioned information types that are too self-evident. We addressed this threat by providing the interview guideline in advance. Concerning external validity, our results are not sufficient for an actual generalization due to the small sample size. However, our study provides a good foundation for further research, e.g., in the form of a quantitative survey and prototypical tool development to improve team collaboration in MSA development.

References

[1] Victor R Basili and H Dieter Rombach. The tame project: Towards improvement-oriented software environments. *IEEE Transactions on software engineering*, 14(6):758–773, 1988.

[2] Mary Jo Bitner, Amy L Ostrom, and Felicia N Morgan. Service blueprinting: a practical technique for service innovation. *California management review*, 50(3):66–94, 2008.

[3] Melvin E Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.

[4] Kim Dikert, Maria Paasivaara, and Casper Lassenius. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, 119:87–108, 2016.

[5] Jonas Fritzsch, Justus Bogner, Stefan Wagner, and Alfred Zimmermann. Microservices migration in industry: intentions, strategies, and challenges. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 481–490. IEEE, 2019.

[6] Valentina Lenarduzzi and Outi Sievi-Korte. On the negative impact of team independence in microservices software development. In *Proceedings of the 19th International Conference on Agile Software Development: Companion*, pages 1–4, 2018.

[7] Nils Brede Moe, Bjørn Haug Dahl, Viktoria Stray, Lina Sund Karlsen, and Stine Schjødt-Osmo. Team autonomy in large-scale agile. In *Proceedings of the Annual Hawaii International Conference on System Sciences (HICSS)*, pages 6997–7006. AIS Electronic Library, 2019.

[8] Sam Newman. *Building microservices*. " O'Reilly Media, Inc.", 2021.

[9] Jonas Sorgalla, Philip Wizenty, Florian Rademacher, Sabine Sachweh, and Albert Zündorf. Applying model-driven engineering to stimulate the adoption of devops processes in small and medium-sized development organizations: the case for microservice architecture. *SN Computer Science*, 2(6):459, 2021.

[10] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.