The Process of Microservice Development: Exploring the Unknown

Jonas Sorgalla

University of Applied Sciences and Arts Dortmund, Dortmund, Germany
jonas.sorgalla@fh-dortmund.de

1 Introduction

A Microservice Architecture comprises multiple microservices which are tailored around business capabilities and should be owned by a cross-functional team covering the whole microservice lifecycle of design, development, deployment, and maintenance [2]. As such, organizations developing a Microservice Architecture (MSA) [6] subject to Conway's Law [4] which states that the system structure is an image of the communication structure of an organization. This naturally puts the development process of microservices in the direction of a *large scale agile* process in which multiple agile teams collaborate to create a system consisting of multiple microservices.

There are various frameworks to manage such scaled agile scenarios, e.g., Scrum@Scale [9], Scaled Agile Framework (SAFe) [1], or the Spotify Model [8]. However, many of these process models are aimed at organizations with at least 50 developers and provide means to manage huge numbers of developer [3]. Therefore, we were curious to see how small and medium-sized companies organized the development of their Microservice Architecture especially given the fact of Conway's Law. We conducted a small qualitative study that included five in-depth interviews with software architects of such companies. Our goal was to generate insights and hypotheses which unique challenges and solutions these smaller companies face during microservice development and how the scientific community can remedy but also learn from them.

2 Case Study

In the following, we briefly describe our case study comprising the applied methodology (cf. Subsection 2.1) as well as the cases and results (cf. Subsection 2.2).

2.1 Methodology

To explore the development process we applied the Grounded Theory Methodology (GTM) [5]. In detail, we explored six different microservice systems and their development processes by talking to five software architects. Each interview partner is currently involved in a leading position and bears at least partial responsibility for the development process. The interviews were done on-site and lasted approx. 1.5 hours each. We performed semi-structured interviews centering around the organization of the development process comprising applied methodologies, tools, and communication patterns. We have also collected basic information about the developed systems in order to better assess the processes. Each interview was than transcribed, paraphrased and coded iteratively.

2.2 Results

Table 1 shows an overview of the six explored development processes. At the time of the interviews, all developments were in the middle of the process and by no means finished. Interviewee I4 reported about two projects he is involved in. The interviews were conducted in the period from autumn to winter 2019.

No.	Case	Project Type	Domain	Services	Devs	Teams
I1	C1	Templated Greenfield	Public Administration	60	≈30	5
I2	C2	Migration	B2B E-Commerce	8	10	3
I3	C3	Greenfield	IoT	18	28	2
I4	C4	Migration	B2B E-Commerce	34	≈10	2
	C5	Migration	B2C E-Commerce	8	≈10	2
I5	C6	Templated Greenfield	Logistics	15-20	75	≈10

Table 1: Case Overview of explored development processes.

Process Model In each process the teams internally apply Scrum [7]. However, on the higher level of collaboration across team boundaries, only I2 describes their collaboration process as belonging to a particular formal model, in this case *Holacracy*[1]. In all other processes a customized large-scale approach is applied and constantly adapted. Especially I4 differs probably because the process emphasizes reusability of services across customized instances of their architecture. During the analysis, we found that knowledge of the overall architecture is distributed among only a few people, which seems to us detrimental to good alignment. However, the overall communication of team activities across team boundaries is neglected in the observed approaches, as this would lead to *I6: "overlong and inefficient meetings"*.

Developer Skills In each case, a high entry barrier as a developer is reported because of the complex environment. All interview partners state full stack knowledge as hiring requirement for the involved developers. However, the interviewees also report that every developer has a certain focus which in all cases centered around front- or backend. Interestingly, each development process has a dedicated unit, in case of C1 and C6 even a complete team, which is solely focused on providing and maintaining infrastructural requirements, e.g. managing Kubernetes. This trend for specilized units might contradict a team's feeling of responsiblity for its services which is a distinct characteristic of MSA [2].

Tools and Techniques With regard to the tools used, all respondents rely on Git, Ticketing and a Wiki, whereby the Wiki is indicated as being neglected by all partners. Instead, all companies stress the use of Swagger[2] to document the interfaces of their MSA whereby Swagger's code generation functionality is not adopted. I2 also emphasizes the use of Terraform[3] to better manage the infrastructure. Graphical modeling such as UML does not play a significant role in any development process, as it is perceived as difficult to keep it up to date and inefficient (I2: *"in this situation, it is not worth the effort"*).

[1]https://www.holacracy.org/
[2]https://swagger.io/
[3]https://www.terraform.io/

3 Conclusion

In our talk we plan to discuss the explored processes, insights and derived hypotheses with the audience. Despite only six cases we achieved a high saturation in most observed categories, i.e., each interview partner reported relatively similar experiences, solutions and challenges. Nevertheless, some additional interviews should be conducted to ensure the quality of the analysis.

References

[1] Achieving business agility with safe® 5.0. Technical report, Scaled Agile, Inc.

[2] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software*, 33(3):42–52, 2016.

[3] K. Conboy and N. Carroll. Implementing large-scale agile frameworks: Challenges and recommendations. *IEEE Software*, 36(2):44–50, March 2019.

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

[5] Barney G Glaser and Anselm L Strauss. *Discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.

[6] Sam Newman. *Building Microservices*. O'Reilly Media, 2015.

[7] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[8] D. Smite, N. B. Moe, G. Levinta, and M. Floryan. Spotify guilds: How to succeed with knowledge sharing in large-scale agile organizations. *IEEE Software*, 36(2):51–57, 2019.

[9] Jeff Sutherland. The scrum@scale guide version 2.0. Technical report, Scrum Inc.