Topic 2 – "On microservices and the missing patterns"

Wouter Joosse
7a4e45277ef
wouterjoosse@gmail.com

***Abstract*— Although microservices are rather popular, research is still somewhat lacking in certain area's. One of those area's is the use of architectural patterns in microservice design. This essay will provide three questions that can be researched in order to help understanding what patterns could be used in designing microservices.**

I. Introduction

In 2014 Fugetta and Di Nitto [1] published a roadmap for research on software process which gave a review of research over the first decade of the 21st century. They identified several trends in the IT market on which researches should focus during the following years, which where mainly hovering around the idea that internet was becoming the driving factor of software development, and that it has made the software universe more diverse.

One of the conclusions was that researchers should be open minded about trends within the software development industry. Around the time that this article was released, a new trend was coming up quickly: microservices. Since the introduction of microservices in 2014 by Fowler and Lewis [2], they have been adopted and used in operations by several large companies, such as Netflix, Amazon and LinkedIn.

The microservice design is an architectural style where small (micro) services are structured around business functions, where each service runs on its own and communicates with other services via a lightweight communication channel. Each service functions as a separate program and only communicate with other services via the communication channel. Other dependencies should not exist. This architecture can be seen as an extension of SOA (Service-Oriented Architecture). [3]

Despite the popularity of microservices, a mapping study in 2017 [4] showed that there are several research area's concerning microservices which are not covered by recent research. One of these area's is the use of architectural patterns in microservices. Out of 71 investigated papers, only 2 where concerned with the patterns used when designing microservices.

This essay will outline three questions that can be asked and researched in order to help mitigating this gap in knowledge.

II. Ideas

Since microservices are popular, it is important to understand what factors lead to success in designing and implementing an microservice based architecture. The following questions will be helpful in determining this:

Q1 Which microservice specific design patterns and antipatterns have been found so far?

This question will focus on getting an overview of previous research on designing and implementing microservices. Some studies have been conducted on best practices[5] and worst practices[6] when designing microservices, and an extensive study was carried out on grey literature (such as: blogs, white papers and video's)[7]. These results, and quite possibly others, need to be consolidated to allow an overview of what to do and not to do when designing and developing microservices.

Q2 What are the costs of antipatterns in microservice design and implementation and what is the frequency of these antipatterns?

Technical dept is a term introduced by [8] to indicate to non-developers what the dangers of shipping sub-optimal code is. Sub-optimal code is code which does what it is supposed to do, but is not "quite finished". For example: it has dependencies which should be removed, or a database which was easier to use in the beginning, but should be replaced by another database. Sub-optimal code or an sub-optimal implementation allows for quick wins, but, in the long run, will drag the software system down.

Technical debt consists out of three parts:

Debt: The existence of sub-optimal components.
Interest: The cost because of debt. For example: a sub-optimal module will cost extra maintenance when the system get's bogged down.
Principal: The cost of repairing the sub-optimal components, i.e. the cost of refactoring.

The notion of architectural technical debt (ATD) [9] has been introduced to denote the kind of technical debt that is associated with architectural structures.

To get better grip on the software development process, it is crucial to know which mistakes are made often, and which mistakes cost the most in terms of resources and money. Operational management can make better informed decisions when this kind of knowledge is publicly available. Since, to the best of my knowledge, no other systematic study has been carried out regarding a categorization of antipatterns in microservice design, a case study must be carried out to provide a glance, albeit limited, on the consequences of antipatterns on technical debt in microservices.

Q3 Which known metrics can be used to detect microservice-specific antipatterns in microservice implementations?

Once the specific antipatterns are identified, a study can be carried out to discover which metrics can be used to signal the existence of antipatterns in microservice code bases. The study can be a combination of a literature survey and a case study, to verify results that have been found during the literature survey.

This study should specify metrics in two different area's. First it will focus on finding metrics specific for code bases of microservice systems, and secondly it will search for metrics that can be used on an architectural level. A previous study [10] has identified several ATD as risks for microservice architecture and can be used as starting point for this.

REFERENCES

[1] A. Fuggetta and E. Di Nitto, "Software process," in *Proceedings of the on Future of Software Engineering*. ACM, 2014, pp. 1–12.

[2] "Microservices," https://www.martinfowler.com/articles/microservices.html, accessed: 2020-01-03.

[3] O. Zimmermann, "Microservices tenets," *Computer Science-Research and Development*, vol. 32, no. 3-4, pp. 301–310, 2017.

[4] P. Di Francesco, I. Malavolta, and P. Lago, "Research on architecting microservices: Trends, focus, and potential for industrial adoption," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 21–30.

[5] D. Taibi and V. Lenarduzzi, "On the definition of microservice bad smells," *IEEE software*, vol. 35, no. 3, pp. 56–62, 2018.

[6] D. Taibi, V. Lenarduzzi, and C. Pahl, "Architectural patterns for microservices: A systematic mapping study." in *CLOSER*, 2018, pp. 221–232.

[7] J. Soldani, D. A. Tamburri, and W.-J. Van Den Heuvel, "The pains and gains of microservices: A systematic grey literature review," *Journal of Systems and Software*, vol. 146, pp. 215–232, 2018.

[8] W. Cunningham, "The wycash portfolio management system," *ACM SIGPLAN OOPS Messenger*, vol. 4, no. 2, pp. 29–30, 1993.

[9] I. Ozkaya, R. L. Nord, and P. Kruchten, "Technical debt: From metaphor to theory and practice," *IEEE Software*, vol. 29, no. 06, pp. 18–21, nov 2012.

[10] S. S. de Toledo, A. Martini, A. Przybyszewska, and D. I. Sjøberg, "Architectural technical debt in microservices: A case study in a large company," in *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE, 2019, pp. 78–87.