Managing Open-source Microservices Projects

Tom Gustafsson[1] and Andrey Saltan[1]

Lappeenranta-Lahti University of Technology LUT, Finland
{tom.gustafsson,andrey.saltan}@student.lut.fi

Abstract. This study intends to understand the project management practices and collaboration principles of participants who contribute to developing open-source microservices projects. Through the lens of Conways Law within this study, we empirically analyze the participation roles of developers with a high number of commits across three projects in GitHub, an open-source software developing platform and repository. We generalize this case study research and hypothesize how the segregation of duties among participants, as well as commits number and distribution, depends on the size of a repository network and the presence of a hub. The paper concludes with suggestions on further investigation of this topic using large-scale research.

Keywords: Open-source · Microservices · Conways Law · Project management

1 Introduction

Since the first open-source software (OSS) development projects were introduced, this phenomenon received sufficient attention in both project management and software engineering research domains. A wide range of studies provides a comprehensive analysis of this novel software development paradigm [2]. Open-sourcing assumes a higher level of freedom and flexibility for project participants, while a number of participants considerably determines success and quality. At the same time, it appeared that there is no silver bullet solution, and a lot depends on the project and team characteristics, as well as the technology used [9]. As a result, a clear and comprehensive answer on how OSS development should be organized with the classification of affecting factors and typology of recommendations is missing.

Back in 1967, Melvin Conway introduced the law, which states that organizations which design systems... are constrained to produce designs which are copies of the communication structures of these organizations [3]. To a certain extent, the critical logic behind this Conways Law is that communication is one of the prime determinants of the system/module development success. A wide range of existing studies aimed to empirically verify Conways law for various types of software development organizations, project setups, development team characteristics. It appeared that there is a variety of opinions regarding what this law encompasses, and studies showed mixed results on its feasibility [1].

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

CEUR-WS.org/Vol-2520/paper4a.pdf

A special issue is the applicability of Conways law to the projects developed in the OSS environment. The ambiguity of the situation is that, on the one hand, it could be incorrect to consider an open community of people involved in an open-source project as a software development organization as it lacked formal structure, goal, and other important artifacts. Similarly, OSS projects with plenty of modules, forks, and modifications are also hardly can be considered as unified systems. However, on the other hand, since Conways law primarily concerns communicative principles rather than formalities, we can expect that the law can work, albeit with certain reservations and updates. While recent studies on these issues started to appear [6], existing literature does not extensively explore this matter and does not provide reliable findings on it.

Within this study, we focus on a particular type of OSS — microservices OSS. Micro-service architecture is the latest trend in software development that promises a sufficient increase in software development agility as each micro-service becomes an independent unit of development, deployment, operations, versioning, and scaling [4]. According to Conways Law, developing a system as a portfolio of microservices in an ideal organization should lead to the situation when most of the developers are distributed across multiple developing teams, each of which works on one micro-service [5].

Within this study, we investigate the project management practices and collaboration principles of participants who contribute to developing microservices OSS projects through the lens of Conways law. We analyze development practices, project-related processes, collaboration patterns of several existing and available on GitHub microservices OSS projects. Further, we generalize the results of our case-study investigation in the form of assumptions that could be further explored using large-scale research.

2 Background

The issue of managing microservices OSS projects as a research area lies at the intersection of three broad research areas: *Open-source software development*, *Microservices architecture* and *Software Engineering and Project Management*. Independently, the research in all these areas is rapidly growing, and there is a diverse range of academic studies that explore topics on the intersection of any two of these areas. However, we could not find any study that touches all three areas. An overview of two studies that form the background for our research is presented below.

The first study investigates project management of Small- and Medium-sized Open-source projects [10]. More precisely, this research focuses on understanding project developer roles that can be divided into two categories: core members and associate members. In this study, interaction frequencies and complete-linkage hierarchical clustering methods were applied to determine which developers were the core members and which developers were the associate members. Based on conducted case-study, this research outline similarity in core members and associate members characteristics.

Another study that was used as a basis for our research explored the applicability of Conways Law to the OSS environment [6]. The prime aim of that research was to provide metrics to verify Conways law for open-source projects, which are already known for the complex nature of developer collaboration networks. The study denied Conways law, which advocated the separation of code and staff into well-isolated groups, which would guarantee efcient code updates, and was, therefore, claimed to be a cornerstone for successful projects. Although Conways Law hardly holds, the examined projects were successful. The research proposed the following explanation to the paradox:

1. There may exist efcient organizational structures alternative to the one than found and explained by Conway;
2. Conways law holds, however, not for module dependencies but task dependencies.

3 Research Design

The overall goal of this research is to obtain a deep understanding of how the development of the microservices OSS project should be organized thought the prism of Conways law. More formally, the following research questions drive our investigation:

- What do we know about project management/teamwork in microservices OSS?
- How are the microservices OSS development teams structured and organized in terms of Conways law?

To answer these two research questions, we conduct an exploratory case-study analysis using data from GitHub on available microservices OSS projects. The data set was collected from three different open-source projects, which are all implemented using microservice architecture. The aim is to find projects which are all using microservice architecture, still different in terms of size and structure.

While [10] proposes a model on how to categorize open source developers as Core and Associate members, within this research, we considered participants as Core members if they are responsible at least for 2% of total contribution to particular micro-service based on GitHub statistics. All the other participants were categorized as Associate members. The aim is to identify how the OSS projects using microservice architecture are organized. We intend to determine if there is a pattern, which can be used for figuring out whether microservices OSS teams are following Conways Law recommendations.

The amount of three chosen OSS projects available for public audits on GitHub is limited. We selected the following three projects that provide enough data for the required analysis: Internet of Things Platform Lelylan[1], Multi-cloud Continuous Delivery Platform Spinnaker[2] and Platform for the Internet of

[1] https://github.com/lelylan/lelylan
[2] https://github.com/spinnaker

Table 1. Characteristics of the involved projects

Project	# of Committers	# of Commits	Changed # of Files	Changed # of Lines
Spinnaker	122	10 800	62 002	1 808 915
Sitewhere	17	2 315	30 449	581 659
Lelylan	7	2 077	13 994	1 840 982

Things SiteWhere[3]. From these three chosen projects, we collected all available data. That includes the changed files, a number of lines added and deleted, timestamps, among others. The collected data set was stored in the SQLite database. Further, we cleaned the data set from the records, which can be seen as erroneous or which were clearly irrelevant to the purpose of our study (i.e., we removed commits by noreply@github.com). SQL queries were executed against the database to get a summary of the history for each Microservice OSS project. The summary included the committer e-mail, a number of changes done by project participants, and the percentage of all the changes in certain Microservice done by each committer.

4 Results

First, we get an overview of project participants involvement across project evolution. We observed the number of microservices within each project and how this number changed over time, how many and in which way project participants have been contributing to the development of these microservices, and what is their distribution based on Core/Associate membership classification.

The largest project out of the three considered is Spinnaker, which had 122 committers doing 10 800 commits, 62 002 file changes, and 1 808 915 changed lines in files. Lelylan and Sitewhere are much smaller projects. Lelylan has seven committers doing 2077 commits, 13 994 file changes, and 1 840 982 changed lines in files. Sitewhere has 17 committers doing 2315 commits, 30 449 file changes, and 581 659 changed lines in files.

While analyzing the Spinnaker project further, it appeared that out of the 122 project participants, 32 could be seen as Core developers at least in one microservice (responsible for at least 2% of all changes to that microservice). When looking this way, each microservice, Core developers are committing at least 80% of all changes in microservice. Venn diagram was drawn from this data, in order to visually see how Conways law is obeyed. In the Venn diagram, one can see that some microservices are not overlapping with some of the others. For example, Echo is not overlapping with Rosco, which means that those microservices have no common core developers in them. While one can see that each microservice is overlapping with some others, one can understand that none

[3] https://github.com/sitewhere/sitewhere

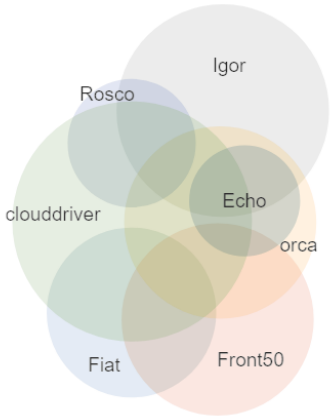

Fig. 1. Intersection of Core developers working on different microservices within the Spinnaker project

of the microservices has a team of core developers that only concentrate on one service as a team. However, in Spinnaker, which is the largest project, there are many areas without overlapping each other, which tells that there some teams core developers of the microservice, which consist of developers who are not core developers of all other microservices (see Fig. 1). That behavior is missing in smaller projects (see Fig. 2), where all the teams had a member, who is a core developer in all microservices.

When analyzing the other two projects, other characteristics were seen. For Lelylan, out of those seven committers, two could be seen as core developers following the rule that core developers must commit at least 2% of all changes, and the core team commits more than 80% of changes. Out of 14 microservices in Lelylan, 13 were done more than 98% by one developer while the same person did 96% of the changes in the last remaining one. Depending on whether the one committing barely over 2% could be seen as a core member, Venn diagram would show 100% 14 or 13 overlapping circles and maybe one of halfly overlapping all the other 13 circles. The same behavior can be seen with Sitewhere, one person would be the core member, and the rest are associated members.

5 Discussion and Conclusion

The prime aim of this research was to investigate the project management practices and collaboration principles of participants who contribute to developing microservices OSS projects through the lens of Conways law.

Results indicate that Conways Law has a lower limit of applicability both regarding project size and maturity. Our findings show that small teams with less than twenty developers, as in two explored cases, do not follow the logic of Conways Law. Core developers in small and newly established projects are

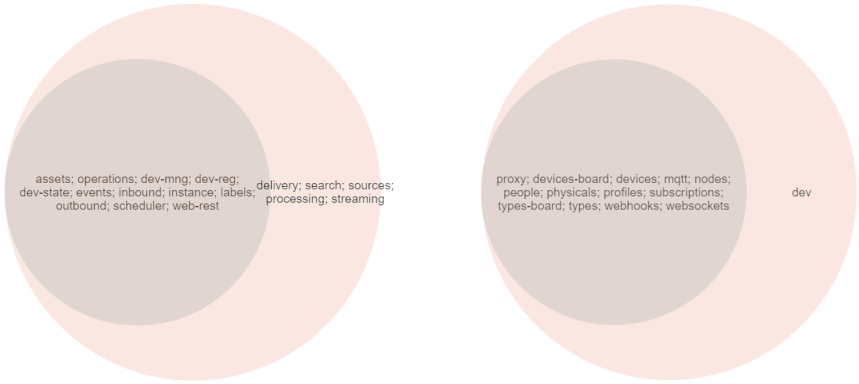

Fig. 2. Lelylan (left) and Sitewhere (right) microservices are both developed by teams consisted of one person and two persons respectevly

working on all microservices within the project. However, when a project is maturing and becoming large enough, it starts to follow the logic of Conways Law. Development teams are becoming more focused on particular microservices, and the overlap between their members is reducing. Still, the full implementation of Conways Law is the OSS environment in general, and in the case of microservices OSS, in particular, is hardly possible. Conways law is not natural, so without efficient project management leadership that is not possible within the OSS environment, the full correspondence of project and organization (team) structures couldnt be reached.

While Conways law states that each team works on its software module, and only on that module [6], it contradicts other research related to large teams. Moore and Spens [8] focused on scaling agile in large software teams. They considered teams with more than 30 developers, and one critical characteristic for successful developers was to ability to operate outside team walls. Also, [7] describes the scalable agile framework in Spotify company, where team size was growing from 25 to 250 in a few years. Much focus was put on communication, information sharing, and freedom to change anything, even though teams were made as independent as possible. It was interesting to find similar behavior naturally happening in open-source projects. Specialized teams start to form, but some core developers are still deeply involved in almost all projects. The example of Spotify highlights that co-located people collaborating and focusing on one area is in place, but so is the idea that anybody interested can join the collaboration.

In a certain sense, the conducted study can be considered as a pilot for an extensive study of the project management practices in misconceives OSS projects.

References

1. Bailey, S.E., Godbole, S.S., Knutson, C.D., Krein, J.L.: A decade of Conway's Law: A literature review from 2003-2012. International Workshop on Replication in Empirical Software Engineering Research, (RESER) Proceedings pp. 1–14 (2013). https://doi.org/10.1109/RESER.2013.14
2. Crowston, K., Wei, K., Howison, J., Wiggins, A.: Free/Libre Open-Source Software development: What we know and what we do not know (2012). https://doi.org/10.1145/2089125.2089127
3. Herbsleb, J.D., Grinter, R.E.: Architectures, coordination, and distance: Conway's law and beyond. IEEE Software **16**(5), 63–70 (1999). https://doi.org/10.1109/52.795103
4. Jamshidi, P., Pahl, C., Mendonca, N.C., Lewis, J., Tilkov, S.: Microservices: The journey so far and challenges ahead. IEEE Software **35**(3), 24–35 (2018). https://doi.org/10.1109/MS.2018.2141039
5. Kalske, M., Makitalo, N., Mikkonen, T.: Challenges When Moving from Monolith to Microservice Architecture. In: Lecture Notes in Computer Science. pp. 32–47 (2018)
6. Kamola, M.: How to verify conway's law for open source projects. IEEE Access **7**, 38469–38480 (2019). https://doi.org/10.1109/ACCESS.2019.2905671
7. Kniberg, H., Ivarsson, A.: Scaling Agile at Spotify with Tribes, Chapters and Guilds (2012),
8. Moore, E., Spens, J.: Scaling agile: Finding your agile tribe. Agile 2008 Conference Proceedings (7), 121–124 (2008). https://doi.org/10.1109/Agile.2008.43
9. Sen, R., Singh, S.S., Borle, S.: Open source software success: Measures and analysis. Decision Support Systems **52**(2), 364–372 (2012). https://doi.org/10.1016/j.dss.2011.09.003, http://dx.doi.org/10.1016/j.dss.2011.09.003
10. Yu, L., Ramaswamy, S.: Mining CVS repositories to understand open-source project developer roles. International Workshop on Mining Software Repositories (MSR) Proceedings (2007). https://doi.org/10.1109/MSR.2007.19