WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Master Thesis

Collaboration networks in open-source software development

Jozsef Csepanyi

Date of Birth: 06.09.1996
Student ID: 11927479

Subject Area: Information Systems

Studienkennzahl: h11927479

Supervisor: Johannes Wachs

Date of Submission: 02. April 2021

Department of Information Systems and Operations, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR INFORMATIONS-
VERARBEITUNG UND PROZESS-
MANAGEMENT DEPARTMENT
OF INFORMATION SYSTEMS AND
OPERATIONS

Contents

List of Figures

List of Tables

Abstract

Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus....

1 Introduction

In recent years open source software solutions have become widely popular and frequently used in both scientific and enterprise use, which can be attributed to a number of factors, most importantly the ease of development and deployment of IT projects, improved cybersecurity and enhanced scalability [9]. This increases the contribution to open source projects from enterprises and individuals alike. Due to its nature, open source software projects are driven by community contributions, and depend heavily on active participation in all phases of the project.

Software development in a corporate environment usually follows a strict hierarchial structure, where each participant is given a precise position and responsibility, like project manager, scrum master, senior or junior developer, and employees do not tend to work outside of their assigned tasks and territories. The main purpose of maintaining software development structures is for the company to ensure that the outcome of the project is in accordance with the business objective, adheres to the pre-set quality criteria and it is completed in a given timeframe; in other words to asses the risks associated with the business objective of the software project [10]. This is achieved by breaking down the developed software into smaller, less complex components, and grouping the developers into managable teams, where the communication is moderated between teams [2].

As opposed to commercial software development, Free/Libre Open Source Software (FLOSS) projects usually do not follow an organizational hierarchy, and are usually self-organizing and dynamic [2]. Issues, bugs and progress are tracked openly, and everyone is encouraged to contribute based on the current topics and expertise, but purely on a volunteering basis. The lack of access restriction to certain modules allows for much more spontaneous interaction between developers, which generate large, complex networks [6]. These complex networks can be seen as large social networks of developers based on collaboration.

Because contribution to FLOSS projects are voluntary, participants have a different motivation for taking part than in commercial software development. According to El Asir et al. [4], FLOSS participation can be motivated by internal and external factors. Internal factors include self-improvement,

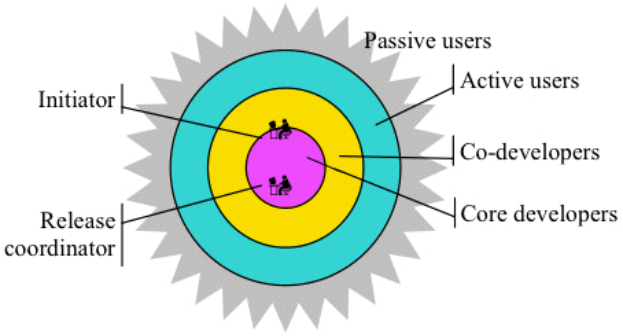

Figure 1: Onion model of collaboration types in FLOSS projects [3].

learning and contribution as a hobby or pass-time activity [1, 11], whereas external factors are motivated by marketing and demonstrating certain skills, thus increasing and improving employability [1].

2 Background and rationale

2.1 Collaboration in FLOSS projects

Collaboration networks of open source software (OSS) have been a subject of many academic research. Raymond [3] has defined collaboration based on bug report interaction, and observed the collaboration network of 124 large-scale SourceForge projects. The generated networks have widely different centralization properties, but it was observed that larger sized projects tend to be more decentralized. The broad community roles contributors tend to take have been also identified in [3], which have been coined as the *onion model* in [6] (Figure 1).

The onion model describes the types of participants in an OSS project as layers. The center represents the small group of core developers, who are responsible for the majority of contributions to the software. They are surrounded by a larger group of co-developers, whose main contributions are usually bug fixes reported by the active users. The passive users are usually the largest in numbers, who do not contribute or report any bugs. In a healthy FLOSS project, each layer of contributors are about one magnitude larger in numbers than the preceeding inner layer [8].

El Asir et al. [4] used a K-means classification to categorise project par-

ticipants into a similar core-periphery structure (core, gray in-between area, and periphery) based on SNA metrics with a montly timeframe, and analysed how and why contributors transition between groups. They found that technical contributions like code commits and lines added have a much heavier impact on becoming a core developer as opposed to other activities, such as testing, reviewing and commenting.

A literature review conducted by McClean et al. [7] systematically analysed the state-of-the-art research of 46 scientific papers in the field of FLOSS social networks, and categorised them into three groups based on topic: structure, lifecycle and communication. They conclude, that the existence of core-periphery structure in OSS projects is well established in the field, which is also an indicator of a healthy FLOSS software. Regarding the lifecycle, generally the core development team does not change significantly over time, however, the project becomes more decentralised and distributed as it matures. A lack of research regarding temporal analyses were identified in the most current knowledge, which was suggested as a future research area in this field.

2.2 Social network of Open-source projects

...

3 Motivation of research problem and research questions

Because there is a high dependency on the community in open source software projects, by understanding how contributions are included and what patterns emerge we can gain valuable insight into the project's current state and its trajectory. As stated before, SNA analysis of OSS have been extensively studied, but there is a lack of research regarding temporal models analyzing the lifecycle of a FLOSS project.

The goal of this paper is to fill in this gap by examining OSS project collaboration networks over time using SNA metrics. More specifically, one part of the research will focus on the evolution of such collaboration networks and comparing and contrasting these networks with the software outcome. The second part will focus on events during a project, and how it affects the

developer collaboration. The research questions, which are broken down into subquestions, are as follows:

1. **How does the temporal lifecycle information of a project influence its success?**

   (a) *Based on temporal models of collaboration, is it possible to predict the outcome of the project?* Since it has been proven that the core collaborators do not change much over the course of the OS software development, our assumption is that any sudden or long-term change, that is not consistent with the other observed projects, can have a significant impact on the outcome (negative or positive alike).

   (b) *Can stages of a FLOSS project with a maturity model be observed?* As most OS software starts with a small collaborator basis and grows over time, it can be assumed, that each project goes through the same steps of open source maturity levels. On the other hand, it is also possible that due to the uniqueness of each project, no such stages are observable.

2. **How do major events in the project lifecycle change the collaboration network of the project?**

   (a) *Do planned or foreseeable events change the collaboration structure?* Major software version releases can be considered foreseeable events of the project lifecycle, which could have an effect on the developer collaboration. For example, there might be a higher rate of interaction between contributors just before a new version is released to clear up the backlog of tasks. But it is also possible, that commit and change rates drop during this time, because the focus shifts to stability and testing instead of new features.

   (b) *How unforeseeable internal or external events affect FLOSS collaboration?* Sudden shocks to the project, such as an announcement of disinterest from major users of the software, discontinued enterprise support of the project, large-scale global events like the pandemic, or sudden employee firings can have significant effect on the core and periphery collaborators alike. By analysing the collaboration network before, during and after such changes, we might be able to recognise patterns, that regularly occur around these events.

3.1 Research methodology

To find answers to the research questions above, first we build a repository analyzer tool, which mines collaboration data from FLOSS projects, generates static snapshot collaboration networks at each given time interval and calculates SNA metrics for each snapshot. Then these metrics can be aggregated over time, or plotted against time to discover changes in the network. The `git2net`[1] [5] Python library provides the necessary tools to mine any project repository that uses git version control. It also incorporates temporal network generation capability, which can be used as a source for creating static collaboration networks aggregated over a given period of time.

We apply a hybrid methodology of qualitative and quantitative research. First, as part of the qualitative research, we choose a small number of repositories to be analyzed. We observe the number of connected components, centrality, number of nodes and mean degree SNA metrics in order to discover the core and peripherial collaborators over the project lifecycles. The basis of collaboration, due to the unavailability of other means of communication, is coediting files. Based on the state of the art research in this field, file coediting proves to be an effective and easy way to represent collaboration between developers.

After discovering the collaboration structure over time, we will match the breakpoints and unexpected spikes or troughs to events within the lifespan of the project. We expect that the key SNA metrics will show a periodicity around planned releases and other reoccurring events (e.g. holiday season). Outstanding values without reoccurrence, on the other hand, are more likely to be consequences of unexpected events. In these cases, it should be observed whether the network is capable of reorganizing itself, or does the event leave a permanent mark on the collaboration structure. A categorization of unexpected events and the level of impact each category has should be observed.

For the quantitative research to be conducted, we will gather a large set of repositories along with major events in its lifecycles. We will then run the miner for all repositories, and with the findings of the qualitative research, we will try to detect all major events and their type (planned or unexpected). We will utilize the `ruptures` [2] library to detect changes in the continuous SNA metrics. If the model is capable to accurately recognise events, then we can also apply it on any repository to detect changes, which will allow

[1] `https://github.com/gotec/git2net`

[2] `https://github.com/deepcharles/ruptures`

us to discover changes in the collaboration network that are not related to publicly known events or releases.

References

[1] Shaosong Ou Alexander Hars. Working for Free? Motivations for Participating in Open-Source Projects. *International Journal of Electronic*

Commerce, 6(3):25–39, April 2002.

[2] Christian Bird, David Pattison, Raissa D'Souza, Vladimir Filkov, and Premkumar Devanbu. Latent social structure in open source projects. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT '08/FSE-16, pages 24–35, New York, NY, USA, November 2008. Association for Computing Machinery.

[3] Kevin Crowston and James Howison. The social structure of free and open source software development. https://firstmonday.org/ojs/index.php/fm/article/download/1478/1393?inline=1, February 2005.

[4] Ikram El Asri, Noureddine Kerzazi, Lamia Benhiba, and Mohammed Janati. From Periphery to Core: A Temporal Analysis of GitHub Contributors' Collaboration Network. In Luis M. Camarinha-Matos, Hamideh Afsarmanesh, and Rosanna Fornasiero, editors, *Collaboration in a Data-Rich World*, IFIP Advances in Information and Communication Technology, pages 217–229, Cham, 2017. Springer International Publishing.

[5] Christoph Gote, Ingo Scholtes, and Frank Schweitzer. Analysing Time-Stamped Co-Editing Networks in Software Development Teams using git2net. *arXiv:1911.09484 [physics]*, November 2019.

[6] Juan Martinez-Romo, Gregorio Robles, Jesus M. Gonzalez-Barahona, and Miguel Ortuño-Perez. Using Social Network Analysis Techniques to Study Collaboration between a FLOSS Community and a Company. In Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, and Giancarlo Succi, editors, *Open Source Development, Communities and Quality*, volume 275, pages 171–186. Springer US, Boston, MA, 2008.

[7] Kelvin McClean, Des Greer, and Anna Jurek-Loughrey. Social network analysis of open source software: A review and categorisation. *Information and Software Technology*, 130:106442, February 2021.

[8] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346, July 2002.

[9] PwC. Leading benefits of open-source software among enterprises worldwide as of 2016. *Statista*, 2016.

[10] Ashish Sureka, Atul Goyal, and Ayushi Rastogi. Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis. In *Proceedings of the 4th India Software Engineering Conference*, ISEC '11, pages 195–204, New York, NY, USA, February 2011. Association for Computing Machinery.

[11] Yunwen Ye and K. Kishida. Toward an understanding of the motivation of open source software developers. In *25th International Conference on Software Engineering, 2003. Proceedings.*, pages 419–429, May 2003.