WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

EFMD
EQUIS
ACCREDITED

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:** 31.March 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 [8]. 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 [9]. 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].

On the other hand, 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 [5]. These complex networks can be seen as large social networks of developers based on collaboration.

Contribution to OSS are voluntary, motivation [4]:

- internal factors: like hobbies or self-improvement [1] or learning [10]
- external factors: employability by better marketing and demonstrating certain skills [1]

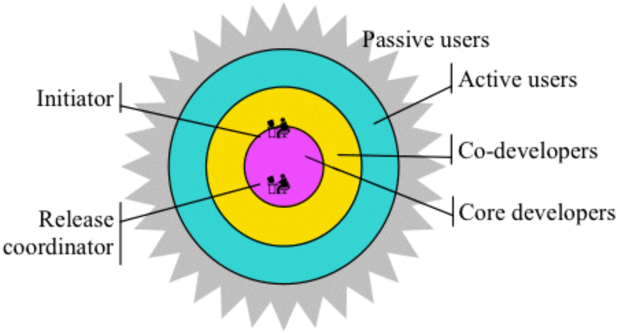

Figure 1: Onion model of collaboration types in FLOSS projects [3].

...

## 2 Related work

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 [5] (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 [7].

El Asir et al. [4] used a K-means classification to categorise project participants 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. [6] 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 are 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.

- Open-source software development properties
  - centralized vs decentralized
- Relevant social aspects of OS projects
- State of the art
  - Problem of analysing changes over time in a network

# 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.

- Importance of OS project analysis based on lit rew
- Analysing effects of large events within the lifecycle of the OS project in order to improve them or adapt
  - Planned, foreseeable changes (e.g. upcoming major release)
  - Unforeseeable changes (e.g. end of support, pandemic)
- Research questions

  - What social patterns emerge within large-scale open-source software projects?
    - Are there smaller "core" collaborator networks connected with weak links or do they form one large interconnected network?
    - Are there usually key contributors, who are central to the project and collaborate with most contributors, or is it completely decentralized?
    - How does the size of the project change these properties?
  - How does the structure of OS software development collaboration change over time?
    - Are there any major changes over the natural project lifecycle? Are they visible in the collaboration network? (e.g. planning, developing, bugfixing, sunset?)
    - How does a sudden major event change the participation and development?
- Preliminary analysis results (pandas, networkx, ...)

## 4 Proposed research method

- Developing a tool, that can extract the collaboration information from any OS project (from GitHub/git repository)
- Data cleaning - method to merge authors (with gambit), excluding common folders, etc...
- Qualitative research
  - Observing collaboration statistics and networks in order to discover patterns: connected components, centrality, changes over time
- Quantitative research
  - Composing a large set of repositories (different sizes, properties)
  - Detecting past changes automatically based on changes in measured statistics

# 5 Outline of thesis

- Literature review
    - Network analysis, relevant metrics
    - Properties of social collaboration networks
- Used repositories, selection criteria
- Data cleaning - files, authors, max modifications
- Implementation
    - ...
- Qualitative analysis
- Quantitative analysis
- Conclusion

## 5.1 (Preliminary literature list - in references)

## 5.2 Work plan including milestones

- Data cleaning - files, authors, max modifications
- Implementation
    - ...
- Qualitative analysis
- Quantitative analysis
- Conclusion

# 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] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.