International Journal of Open Source Software and Processes
Volume 14 • Issue 1

Organizational Influencers in Open-Source Software Projects

Roland Robert Schreiber, Information Systems and Social Networks, University of Bamberg, Germany*

ABSTRACT

Traditional software development is shifting toward the open-source development model, particularly in the current environment of competitive challenges to develop software openly. The author employs a case study approach to investigate how organizations and their affiliated developers collaborate in the open-source software (OSS) ecosystem TensorFlow (TF). The analysis of the artificial intelligence OSS library TF combines social network analysis (SNA) and an examination of archival data by mining software repositories. The study looks at the structure and evolution of code-collaboration among developers and with the ecosystem's organizational networks over the TF lifespan. These involved organizations play a particularly critical role in development. The research also looks at productivity, homophily, development, and diversity among developers. The results deepen the understanding of OSS communities' collaborative developer and organization patterns. Furthermore, the study emphasizes the importance and evolution of social networks, diversity, and productivity in ecosystems.

KEYWORDS

Ecosystem, Evolution, Open-Source, Organizational Influence, Social Network Analysis, Software Development, TensorFlow

INTRODUCTION

In an environment of technological complexity and competitive challenges, software is no longer developed only in-house by a single firm. Instead, there is a collaboration among a community of volunteers, in-house developers, developers from partner companies, universities, and even competitors (Bengtsson & Kock, 2000; Ghobadi & D'Ambra, 2012). Numerous famous firms support OSS projects and cooperate with others (Capiluppi et al., 2012).

This collaboration with competitors has given rise to the concept of "coopetition," a term describing the coexistence of competition and cooperation and the interaction between companies that have a partial congruence of interests (Linåker et al., 2016; Nguyen Duc et al., 2017). There are many examples of open-source initiatives exemplifying coopetition such as WebKit, Blink, OpenStack, and CloudFoundry (Jansen, Finkelstein, & Brinkkemper, 2009; Teixeira & Lin, 2014; Nguyen Duc et al., 2017). Another example of a large-scale programming cooperation is autonomous vehicles. In this context, the experience

DOI: 10.4018/IJOSSP.318400 *Corresponding Author

of all autonomous cars will train every autonomous car and learning from others' failure (Wiseman, 2022). The rate of learning and gaining experience is faster with this approach than when compared to a sole human driver whose only training source is personal experience.

There are, however, increasing software development challenges in this space, including the development of essential innovative products, long geographic distances between participants across different time zones, cooperation, and open software standards (Frischbier et al., 2012; Holmstrom et al., 2006; Ouriques et al., 2019; Xia et al., 2016). In addition, considerable participatory development becomes more important as the complexity of software projects increases (Çaglayan & Bener, 2016; Wohlin et al., 2015). These OSS projects also depend significantly on the effective use of basic software components, interactions among people, and the human factors identified as key to successful software projects (Biçer et al., 2011; Chow & Cao, 2008; Holmstrom et al., 2006; Oliveira et al., 2018; Wohlin et al., 2015; Wu & Tang, 2007).

This paper explores the developers' and organizations' dynamic informal structure in the TensorFlow (TF) OSS ecosystem over all releases. Employing a case-study approach (Basili et al., 1999), we analyze how collaboration works in the open-source space and, specifically, in the TF OSS ecosystem. We also examine the importance of the top-contributing organizations within the TF OSS ecosystem by considering the social relationship between developers and their affiliations with an organization. The longitudinal case study design is based on applying well-known social network analysis (SNA) techniques, which we use to mine development data over five years. Furthermore, we use the dynamic evolution of the TF networks over time to analyze the characteristics and changing collaboration structures for developers and organizations. Thus, we contribute significantly to closing the software research gap by exploring the role of organizations, developers, sub-communities, and the social network for OSS ecosystems (Abufouda & Abukwaik, 2017; Schreiber & Zylka, 2020). The results reveal new and unexplored aspects of the TF OSS ecosystem that increase our knowledge of informal coordination structures and their evolution with increasing organization participation (Gonzalez-Barahona & Robles, 2013). These aspects are crucial for understanding how a large OSS ecosystem works and help to take appropriate actions if problems emerge, e.g., an ecosystem becomes dominated by a single company. Moreover, the results of this paper are relevant for researchers seeking an overview of TF OSS ecosystem collaboration, insights into ecosystem software development, the role of social networks, and opportunities to contribute to research in this specialized field.

The remainder of this paper is organized as follows. We first present the background and related work. We then formulate the research questions and our research focus before we describe the case study methodology, including the data collection and research methods. The following section details the results of the TF OSS project case study and our evaluation of the results before we provide the limitations and then offer relevant summarizing conclusions.

BACKGROUND

A software ecosystem is the interaction of a set of actors with a common technological platform used by several software solutions (Jansen et al., 2013; Manikas & Hansen, 2013). Table 1 offers several examples of OSS ecosystems and cases of open coopetition.

OSS ecosystems evolve from self-organized and dynamic processes in which volunteers and different firms worldwide contribute to a software product (Gerber et al., 2010; Madey et al., 2002). These ecosystems have a tremendous impact on computing and OSS software development (Zhou et al., 2017), and are gaining importance and supply vital software components and infrastructures, such as operating systems, libraries, component stores, and entire platforms (Ghafele & Gibert, 2014; Jansen, Brinkkemper, & Finkelstein, 2009).

Volunteers from different companies collaborate in these OSS ecosystems. The participating companies gain a mutually beneficial competitive edge through this collaboration that allows them to leverage their own software and services (Barbosa & Alves, 2011; Dagnino & Rocco, 2009; Morgan

Table 1. Open-Source Software Ecosystems

Project	Domain	Examples of companies participating in OSS ecosystems
WebKit	Web browsing	Apple, Nokia, Google, Samsung, Intel, BlackBerry
Blink	Web browsing	Google, Opera, Intel, Samsung
OpenStack	Cloud computing	Rackspace, Canonical, IBM, HP, VMware, Citrix
Hadoop	Distributed computing	Facebook, Twitter, LinkedIn, Jive, Microsoft
Linux	Operating system	Fujitsu, HP, IBM, Intel, Samsung, Hitachi, Red Hat
Eclipse	Software development	Actuate, CA, IBM, Google, Oracel, SAP, Red Hat
TensorFlow	Machine learning	Google, Dropbox, Airbus, Uber, Deepmind, JD.com

& Finnegan, 2014). However, coopetitive relationships are complex, difficult to manage, and create organizational conflicts (Tidström, 2009).

Software forges are especially useful for large OSS projects and OSS ecosystems (Cosentino et al., 2017). The platform GitHub represents the newest generation of software forges, combining traditional capabilities (e.g., free hosting and a version control system) with several social features (Guendouz et al., 2015; Squire, 2014). It also supports the Git version-control system, with its features, social interactions (e.g., bug-tracking, issue-tracking, pull requests support, and profiles), and a powerful GitHub API to provide access to metadata around its hosted software projects (Rashid & Prakash, 2022).

Software development in OSS ecosystems highly depends on social aspects such as effective interaction among people, companies, and human factors that have been identified as key to successful software projects (Amrit et al., 2014; Biçer et al., 2011; Holmstrom et al., 2006; Oliveira et al., 2018; Yilmaz et al., 2016). The success of effective and efficient software development in ecosystems continues to be influenced significantly by collaboration in social networks (Latorre & Suárez, 2017). The inadequate exchange of information and communication between developers and users, which is entrenched in social networks, can cause the downfall of software development endeavors (Charette, 2005). This is why these networks should be developed and managed purposefully to enhance the effectiveness and efficiency of software development (Pryke & Smyth, 2012). For this reason, it is essential to carefully construct, manage, and analyze these networks to maximize software development productivity (Hinds & Lee, 2009).

These social aspects are the fundamental aspect of the investigation in this research study. Social network analysis (SNA) of OSS ecosystems has the potential to reveal hidden structural issues, top influencers, and collaboration patterns (Fischbach et al., 2009; Šmite et al., 2017), knowledge of which can aid in ensuring success.

There is considerable research into the static features of the communication networks of community members and the structural characteristics of developer collaboration networks in OSS ecosystems. The development of software in large OSS ecosystems is knowledge-intensive, and the interactions between the members are complex and self-organized (Behfar et al., 2018; Shah, 2006). Research to date has focused on social project structure and clique analysis, which includes the topics of core periphery (core team and enhanced team) and cluster (Concas et al., 2017; El Asri et al., 2017; Joblin et al., 2017; Toral et al., 2009; Yu et al., 2016) in particular. Other findings support that cohesive social ties between team members in their social networks leads to more productivity (Lee et al., 2013; Singh et al., 2011). However, research on the dynamics of social networks in software development ecosystems over the entire course of the ecosystem lifecycle is missing. In addition, there is a lack of work on forming subgroups and their behavior within software development ecosystem communities (Herbold et al., 2021; Schreiber & Zylka, 2020; Seker et al., 2022).

We examine the TensorFlow (TF) OSS ecosystem community as a case study for investigating the developer and organizational collaboration network. The TF ecosystem is an important, mature,

and established open-source community in GitHub (The State of the Octoverse, 2021). Furthermore, TF is a software ecosystem with standard elements and libraries to help develop and train large-scale machine learning (ML) models based on industrywide and open-source standards.

TF is an end-to-end platform for ML with a comprehensive, flexible ecosystem of tools, libraries, and a strong community. It is a structured ML platform that provides a variety of built-in capabilities, and supports add-on libraries and APIs for key company concerns, such as model building, development of deployment pipelines, and powerful experimentation. Its main feature is that it allows developers to create and deploy state-of-the-art ML-powered applications. The OSS ecosystem was initiated by Google Brain and released under the Apache 2.0 open-source license in November 2015 to accelerate its evolution with the support of a larger community. There are about 3,000 developers involved in the TF ecosystem, 86 subprojects, 3 million lines of code, and 161,000 dependent repositories. It has many supporters, including Google, Dropbox, Airbus, and JD.com. Moreover, TF is a popular and dominant ML framework in commercial production (Dinghofer & Hartung, 2020; Han et al., 2020).

RESEARCH QUESTIONS

Our research focus is the TF coding-collaboration relationship between developers and the organizations for which they work. We analyze the different forms of collaboration, rivalry, competition, and cooperation that take place within the OSS development of the TF ecosystem (Table 2).

METHODOLOGY

Our case study combines qualitative analysis of mining software repositories and the use of SNA on publicly available data from the TF OSS ecosystem. Figure 1 summarizes the study design.

Data Collection

The entire project we explored comprises three principal projects—tensorflow (basic library of TF, written in small letters), docs (documentation), and community (project for documents used by TF developer)—and 86 subprojects. We then mined the code and logs of software repositories to obtain deeper insights into the community. The git system records all commits of the ecosystem, with comments, and documents them in the changelog. Therefore, we extracted the relevant information from the changelog for our study (Figure 2). We then connected the developers and firms that worked on the same files and constructed a social network based on the collaboration patterns.

Research Method

During this research, we constructed the collaboration network by going through the multiple stages of the primary projeact tensorflow. Our data, drawn from the extraction process, spanned the period of November 9, 2015, to July 24, 2020. TF is a fast-moving, community-supported OSS ecosystem.

Table 2. Research Questions

No.	Research questions
RQ1	How does collaboration in the TF OSS ecosystem evolve?
RQ2	Which organizations contribute to the ecosystem?
RQ3	Which organizations collaborate in the TF OSS ecosystem?
RQ4	Is there a tendency toward clustering among developers from the same company (homophily)?

Figure 1. Overview of the Study Design

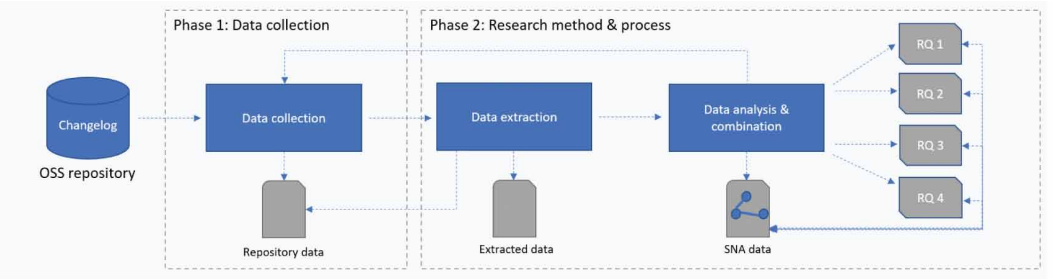
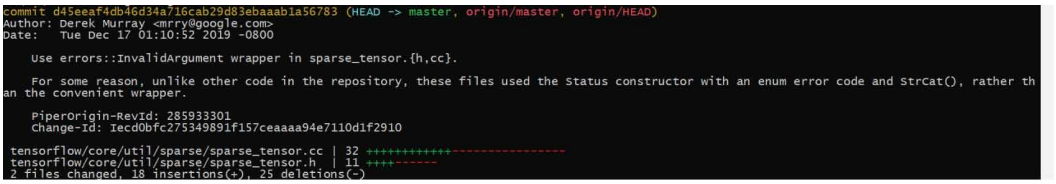


Figure 2. Example of a Commit Record



Initially, the developers were distinguished by email address and were assigned a company if indicated in the changelog records. Depending on the subsequent evaluation, a node represents a developer or a company in SNA. There is a collaboration when different developers edit a file together, designed as an edge between different nodes.

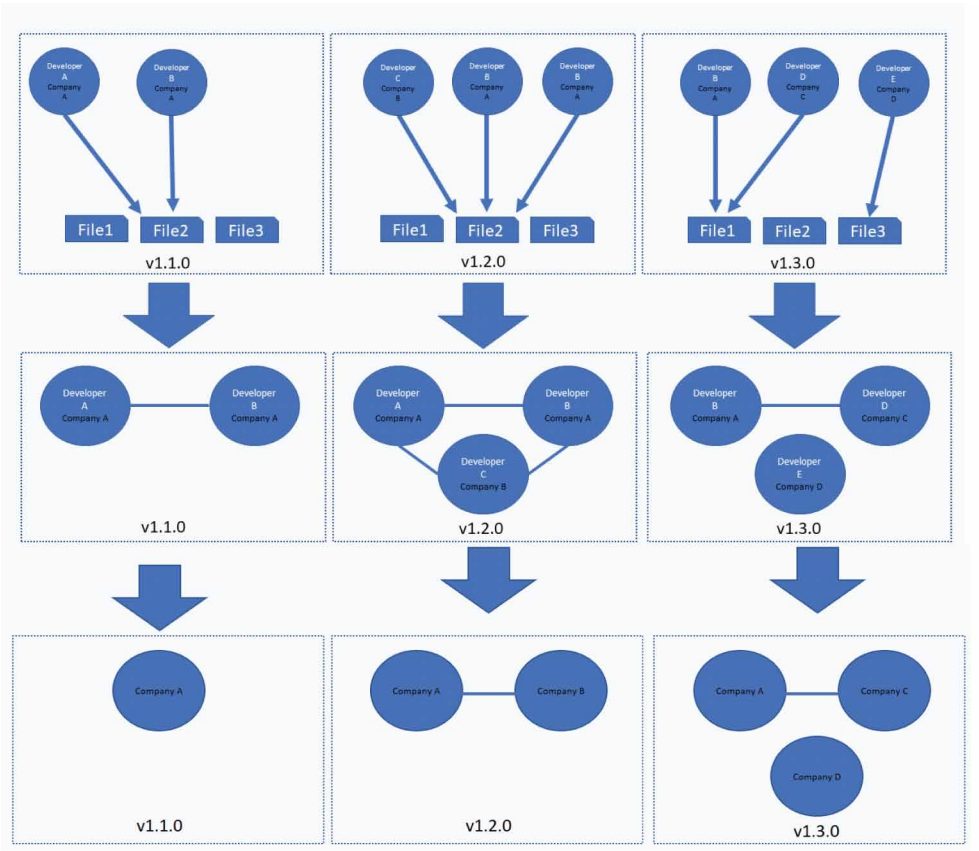
Figure 3 shows the specific construction process of the network of developers and companies, with the example of different release versions and distinct developers. The different developers A, B, C, and D were employed by various software development firms, such as Company A, Company B, or Company C. Every new version is treated as an independent knowledge product produced by the developers who contributed to the code for that version. For a collaborative relationship between two developers to be established, they must both have had an active role in the release of the same version of the source code. That allows the construction of different collaboration networks of developers and companies for each time slice. In Figure 3, there is an aggregation in version 1.2 from the developers' to the companies' network because Developer A and Developer B are employed by Company A while Developer C is employed by Company B.

Based on this method and the extracted relationships, we could construct the social network at a clearly defined time (Figure 3). Finally, we assembled the entire social network for SNA at different evolution stages by time and release dates.

In line with existing guidelines on combing digital trace data with SNA (Howison et al., 2011), we constructed the different collaboration networks of developers and companies for each time slice to examine the evolution of social networks in TF ecosystems. That way, we could assess how the collaboration has evolved and uncover interesting patterns. We analyzed the social network data and visualizations with Gephi (v0.9.2) (Bastian et al., 2009) and the plugins MultiMode Networks Transformation, Node Color Manager, and Groups by partition. Table 3 lists the major releases of TF in the main project tensorflow addressed in this study.

Furthermore, we applied different SNA methods to investigate the constructed networks more deeply. As a result, we can describe social networks through rational and structural characteristics. The rational dimension focuses on the links between pairs of individuals and can be described in

Figure 3. Construction Process for the Developer Network



terms of homogeneity. Table 4 lists the basic social network properties used in our investigation and the dimensions we encountered in our case study.

Based on these basic social network properties, many social network theories provide additional perspectives on complex social ties in ecosystems. We identified relevant SNA theories for our research, which are summarized in Table 5.

RESULTS

An Overview of the Developer Collaboration Over Time

In this section, we address RQ1 for an overview of the developer collaboration evolution over time in the TF ecosystem. We used archival history data from the source version control system and covered more than five years (2015–2020) of the ecosystem's life. The research is based on the main versions of the TF OSS.

The entire TF ecosystem works with pull requests to ensure the project quality and goals, meaning that one TensorFlow team member will check the pull request with code changes and approve it. After the approval, the pull request is merged automatically on GitHub. Final releases are tagged with version labels (e.g., v1.1.3). After that, the master branch of TF is primed for further feature development and bug fixes to be implemented.

To get a better view of the developer collaboration over time, we analyze structural social network metrics: the sum of the relevant developers, network density, and the number of communities. We

Table 3. Major and Minor Releases from Tensorflow of the TensorFlow Community

Release name	Release date	Release name	Release date
v2.3.0	24.07.2020	v1.4.0	01.11.2017
v2.2.0	05.05.2020	v1.3.0	17.08.2017
v2.1.0	07.01.2020	v1.2.0	14.06.2017
v1.15.0	14.10.2019	v1.1.0	21.04.2017
v2.0.0	27.10.2019	v1.0.0	11.02.2017
v1.14.0	19.06.2019	v0.12.0	20.12.2016
v1.13.1	25.02.2019	v0.11.0	09.11.2016
v1.12.0	02.11.2018	v0.10.0	08.09.2016
v1.11.0	25.09.2018	v0.9.0	21.06.2016
v1.10.0	08.08.2018	v0.8.0	22.04.2016
v1.9.0	09.07.2018	v0.7.0	16.02.2016
V1.8.0	27.04.2018	v0.6.0	10.02.2016
v1.7.0	29.03.2018	0.6.0	10.12.2015
v1.6.0	28.02.2018		
v1.5.0	25.01.2018		

Table 4. Basic Network Properties

Structural characteristics	
Social Network Property	**Definition**
Density	The number of direct ties in a network as a ratio of the total number of possible links (Wasserman & Faust, 1994).
Size	The sum of relevant actors (Wasserman & Faust, 1994).
Degree centrality	The number of direct ties to other nodes (Scott, 2013).
Diversity	Simpson's Diversity Index shows the diversity of the open-source developer community's diversity based on developer associated organizations (Jiang et al., 2018).
Rational characteristics	
Concept	**Definition**
Homogeneity	The similarity of nodes (Schenk, 1995).

Table 5. SNA Theories

Theory concepts	Definition
Clique analysis	This analysis of social structure focuses on how connections between large social structures can be built with small and tight components (Kappelhoff, 1987).
Embeddedness	Trusting relationships between actors tend to expand through broker exchange. Trust acts as the primary governance structure in cooperation (structural effect of transitivity) (Uzzi, 1997).
Structural holes	Individuals hold certain positional advantages or disadvantages based on how they are embedded in social structures. A structural hole is a gap between two nodes with complementary sources of information (Burt, 2009).
Power-law distribution	A few actors have many incoming social ties, and predominant actors have just a few ties (Barabási & Albert, 1999).

also show the number of commit activities without merge commits to represent the developer's community productivity.

Figure 4 illustrates the number of actively contributing developers (nodes) over releases. It is observed that the amount of actors involved in development increases with each new software release, signifying increasing developer engagement.

The TF ecosystem network density decreased nearly steadily with increasing developer community and collaborations during the ecosystem lifetime, as Figure 5 shows. This decrease shows that the number of direct ties in a more extensive network decreased over the different TF software releases.

Figure 6 shows the increasing number of communities present throughout successive TF releases. The network remained segregated starting with the initial two releases, and we observed numerous adjacent subgraphs sprouting. Over the TF development period, an enlarged major component has emerged gradually.

The number of commit activities without merge commits represents the increasing developer activity. Figure 7 illustrates that the longer the ecosystem lives and the more developers collaborate on them across different versions, the greater the number of commits. In summary, this means the number of commit activities without merge commits can represent the entire increasing productivity of the TF ecosystem.

Figure 4. Number of Contributing Nodes over Releases

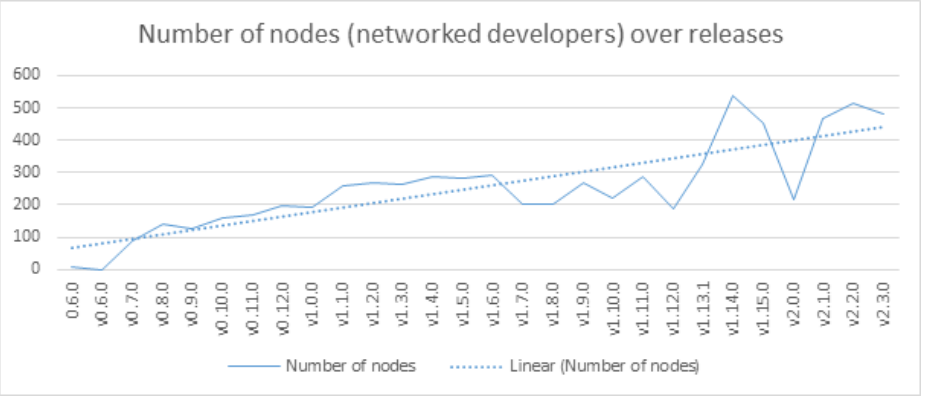


Figure 5. Network Density Over Releases

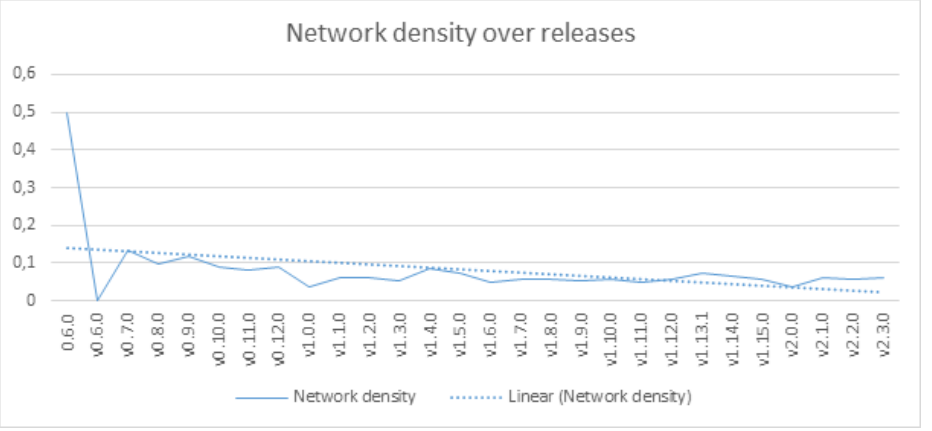


Figure 6. Number of Communities Over TF Releases

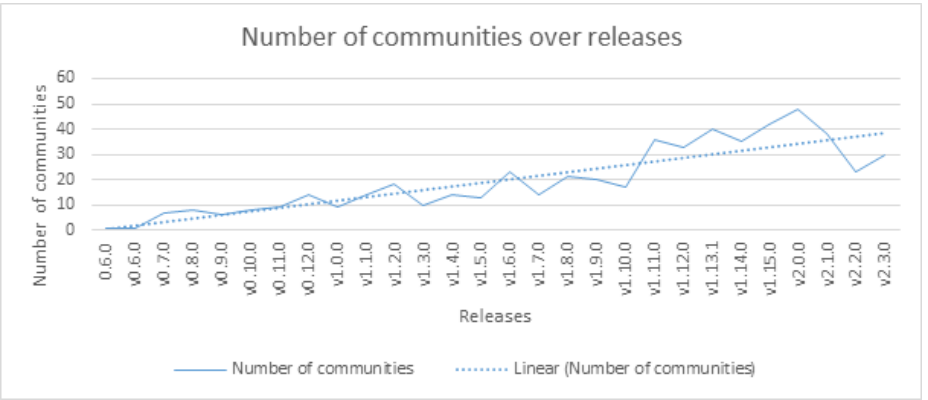
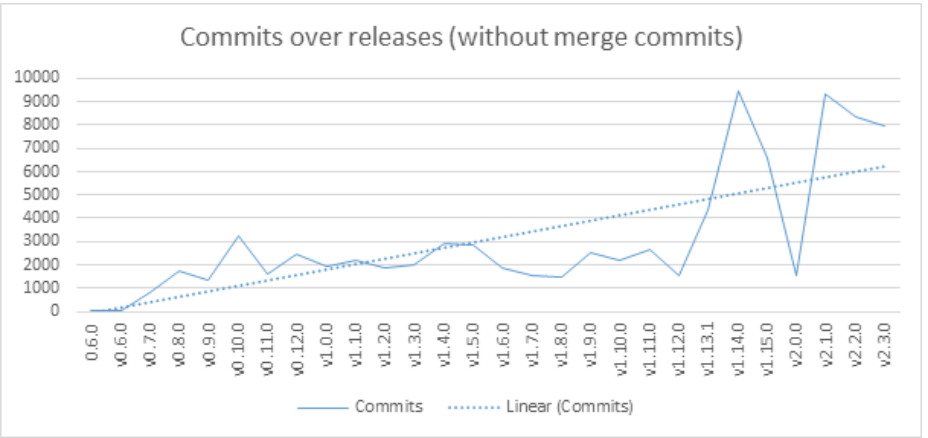


Figure 7. Commits Over Releases (Without Merge Commits)



Identify Top Organizations that Contribute to the TF Oss Ecosystem

To address RQ2, we will investigate the relationship between the developer community and its affiliated organization. The Simpson's Diversity Index was employed to better understand the TF ecosystem's diversity.

Figure 8 shows that Simpson's Diversity Index increases over the different versions. The greater the diversity index, the greater the diversity of the open-source developer community and its associated organizations (Jiang et al., 2018). The dominant organizations decrease, and there is an increasing diversity of organizations over the releases. A high level of diversity shows a healthy software ecosystem (Silveira & Prikladnicki, 2019; Mens & Grosjean, 2015; Vasilescu et al., 2015). The latter is consistent with general research findings that larger open-source developer communities are more diverse than smaller communities (Jiang et al., 2018).

As a general result, the TF ecosystem demonstrated rising productivity with increased diversity and commits in the TF ecosystem community (Figure 9). Furthermore, as demonstrated in Figure 9, the developer community and diversity expand over releases. Therefore, it is apparent that the overall productivity of the TF OSS ecosystem is growing.

Our findings reveal that 20 organizations play key roles in contributing to TF source code. Table 6 reveals these firms.

Figure 8. Simpson's Diversity Index Over Releases

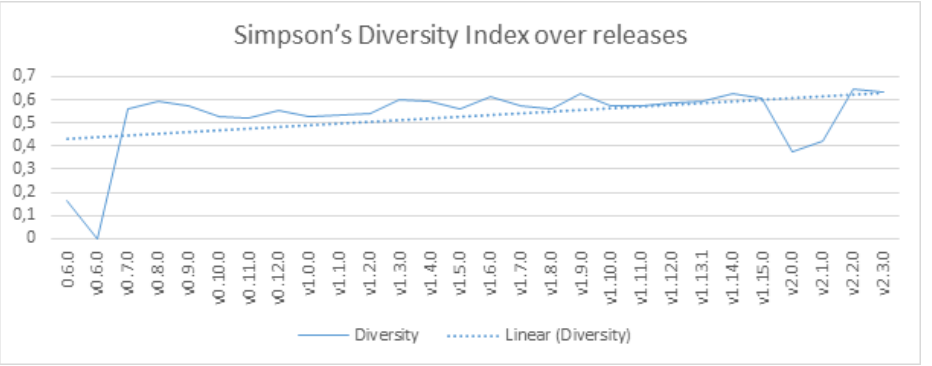
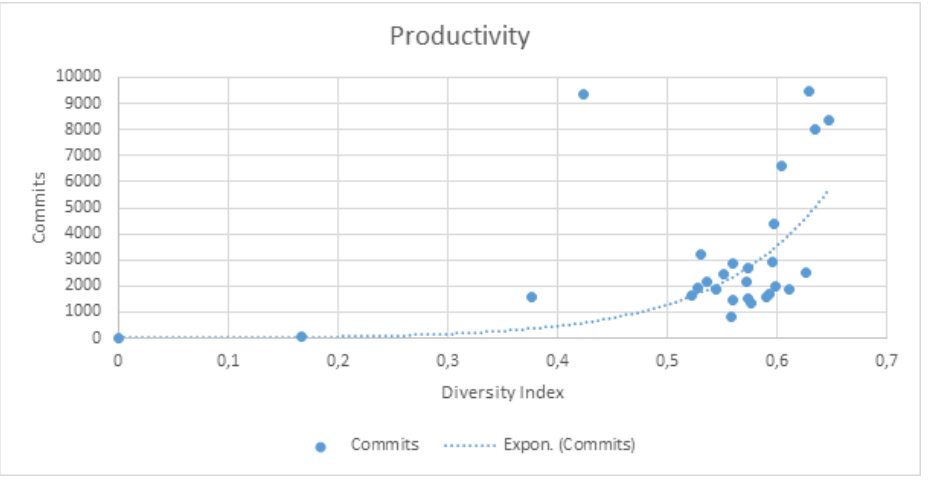


Figure 9. Productivity - Commits and Simpson's Diversity Index



The Organizations Collaborate Over Time in The TF Ecosystem

To answer RQ3, we analyzed the organizations' collaboration networks over the releases of the TF ecosystem. The visualizations in Figure 10 show how key players collaborate in the TF software ecosystem.

In Figure 10 the SNA visualizations illustrate the organizations' code collaboration using combinations of color, size, and location information (Scott & Carrington, 2011). Each node represents one organization with at least one developer. A direct link shows that in the release history, one developer from a different organization edited code in the file previously written by the other developer. The node colors are based on Table 6; other organizations were colored gray. The size of a node depends on its degree-centrality; the larger the node, the more code-collaboration connections the organizations have. The higher an organization node's degree-centrality, the more it is collaborating with others. The topology of the network is centralized around the two organizations (Google and Tensorflow), whereas the other organizations are changing across the individual releases. Furthermore, the number of collaborating organizations is changing over the releases. The top organizations and their developers are responsible for most code changes within the TF OSS ecosystem; this emphasizes the importance. Here, a power-law distribution is recognizable. Google dominates the network initially; a wide variety of organizations are included in the various releases. This leads to more external input and a more dynamic TF OSS ecosystem community.

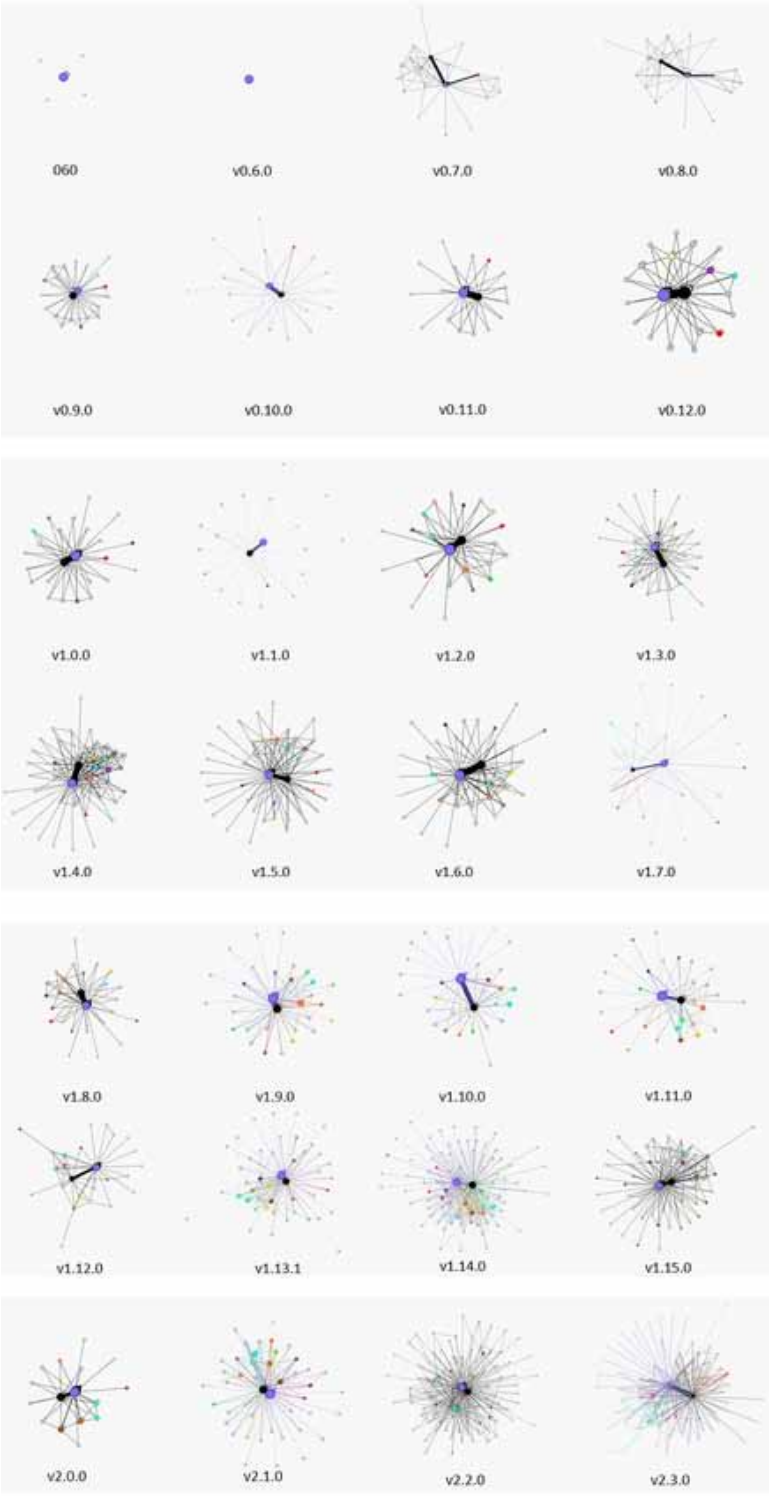
Table 6. Top 20 Firms Contributing to the TF OSS Ecosystem

Organization	Firm description	Color
Alibaba	Alibaba Group Holding Limited is a Chinese multinational technology company.	
Amazon	Amazon is a U.S. multinational technology company.	
AMD	AMD is a multinational semiconductor company based in America.	
Ant Group	Ant Group is an affiliate company of the Chinese Alibaba Group, which owns Alipay.	
ARM	ARM is a British semiconductor and software design company.	
Codeplay	Codeplay, with headquarters in Scotland, creates different software solutions.	
Entropy Source Consulting	Entropy Source Consulting, based in the United States, is a consulting company.	
Facebook	U.S.-based Facebook is a technology conglomerate corporation.	
Google	Google is a U.S. multinational technology company.	
Graphcore	Graphcore is a British semiconductor company.	
Huawei	Huawei is a Chinese multinational technology company.	
IBM	IBM is a U.S. multinational technology, software, and consulting company.	
Intel	Intel is a U.S. multinational technology corporation.	
Linaro	Linaro is a British engineering organization.	
Microsoft	Microsoft is a U.S. multinational technology, software, and consulting company.	
MobileIron	MobileIron is an American software company for mobile devices.	
Nvidia	Nvidia is a U.S. multinational technology and software company.	
QuarkWorks	QuarkWorks is a U.S. consulting company that focuses on software programming.	
Tencent	Tencent is a Chinese multinational technology conglomerate holding company.	
TensorFlow	TensorFlow is an open-source community.	

Tendency Towards Clustering from Developers of the Same Organization

In theory, most organizations' social homogeneity creates strong baseline homophily in networks formed (McPherson et al., 2001). However, the visualizations presented in Figure 10 uncover the different collaborative network structure results of the TF OSS ecosystem. In addition, the sub-

Figure 10. Topologies of the Organization's Collaboration Network Over Different Release Versions



community detection process revealed a small degree of homophily in code collaboration. These organization collaboration networks are highly heterogeneous over the different TF releases.

The Figure 11 SNA visualizations illustrate the developers' code collaboration using combinations of colors for organizations and location information. Each node represents one developer associated with one organization. A direct link shows that in the development history, one developer edited code in the file previously written by another developer in the same software release. Thanks to the TF ecosystem's openness, companies can access each other's resources and share developer costs (Bengtsson & Kock, 2014; Teixeira & Lin, 2014). Figure 11 shows that the sub-communities are highly heterogenous, including developers from many firms. While Google and TensorFlow dominate early versions, new organizations gradually joined the network. In addition, developers often collaborate with peers affiliated with competing firms. For example, Intel and Microsoft work together on the TF OSS ecosystem in specific development release cycles. This collaboration leads to more external input, higher diversification, and a more dynamic TF OSS ecosystem community.

LIMITATIONS

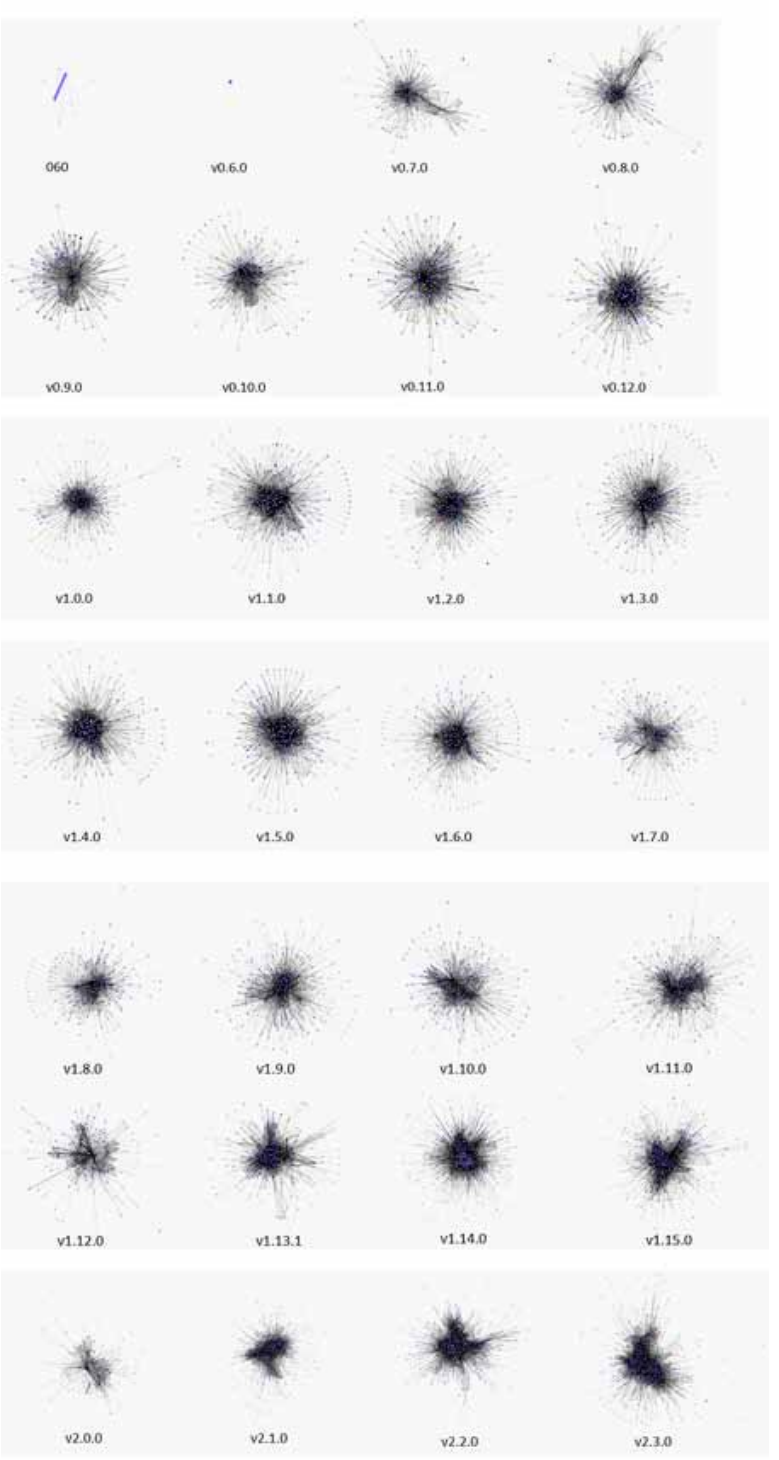
This work analyzes the importance of top-influencing organizations within TensorFlow's software ecosystem by considering the developers' social relationships. Based on a case study approach (Runeson & Höst, 2009), we explore how organizations collaborate in the open-source space and how complex open-source ecosystems, such as TensorFlow, operate. The scheme of validity and threats distinguishes between construct validity, internal validity, external validity, and conclusion validity (Runeson, 2012; Wohlin, 2012).

Construct validity refers to how the operational measures studied represent the phenomena we investigated. One problem is that the collaboration was narrowed to working on the same source code files in the TF GitHub repository and excluded other software development activities, such as testing, code-review/code merging, design, and specification. In addition, we counted only modifications to the same file made during a specific software release as collaboration. The developer's company email address identified the individual developer, which also determined company affiliation. In addition, commissioned third-party developers of a software supplier firm could not be assigned to the original company.

Regarding research questions, we defined five metrics to get a deeper view of developer collaboration. To address this, we used basic quantitative structural network characteristics like the number of contributing nodes, the network density, and the number of communities over the different TF releases to analyze the TF developer community. Initially, the contributing nodes metric shows the number of actively contributing developers over releases and is useful to show trends in the TF developer community. We further relied on the network density that affects the number of direct ties over different software releases. To operationalize the concept of core and enhanced developers, a key measure is the number of communities. Furthermore, the number of commits without merge commits should be considered when examining the developer activity. By applying and evaluating these SNA metrics, it is possible to get an overview of the developer collaboration over time and to achieve a solid basis for SNA visualizations. In addition, applying SNA to developer networks suggests that the metrics are reliable and valid (Meneely & Williams, 2011). Moreover, we used Simpson's Diversity Index to analyze deeper the TF developer community for diversity and to apply an established indicator of the healthiness of the software ecosystem (Silveira & Prikladnicki, 2019). Furthermore, our methods were supported by visualizing the results using SNA graphs; other visualizations and measures could be helpful.

Regarding the internal validity of our study, the results could be inaccurate if we have misinterpreted the SNA visualizations and other metrics. We carried the process out with various measurements and implemented different methods to minimize this limitation. To prevent bias in the results, we endeavored to make the entire research traceable and straightforward in this paper.

Figure 11. Collaboration Network in Different Release Versions with Developer Nodes Associated with 20 top-Influencing Organizations



External validity concerns the degree to which the results of a study can be generalized (Yin, 2014). This research focuses on a single case study of a large open-source software development ecosystem. While it is possible to generalize the results to other software development ecosystems, the external validity of findings is reasonable, and further investigation is required to explore other collaborations related to OSS.

We were intent on developing conclusion validity that would enable us to draw the correct conclusions about relationships in data. In addition, in our study, we were concerned with the ability to replicate our findings. Through a combination of SNA visualizations and other measures, we sought to achieve a balance that would provide an accurate perspective of the research results. To ensure accuracy, each step in the case study was rigorously validated, and researchers regularly carried out assessments.

CONCLUSION

Through a longitudinal study of the TF OSS ecosystem for over five years, this research explored the collaboration of developers and organizations' coding activities. We used a combination of SNA methods and mined data from the TF software repository to gain insight into structural evolution. As a machine learning platform with an ever-expanding range of functionalities, TF necessitates significant maintenance resources and presents an opportunity to leverage the collective strengths of the open-source ML community.

The outcomes show that with the network size expansion (number of nodes), the number of communities increases over the TF releases. Because of these trends, the commit number and the changed source lines are also rising. This trend decreases the network density, which means many clusters connect. For a deeper understanding, we examined the developers and their affiliations with an organization, using Simpson's Diversity Index over the evolution of the TF ecosystem. Data collected from the index shows that the ecosystem will experience an expansion in diversity throughout its lifespan as developers become more abundant. As a general result, with increasing diversity and commits to the ecosystem, the entire TF ecosystem's productivity increases. Furthermore, our findings reveal 20 top-influencing organizations are contributing to the source code. These companies are partly in a competitive relationship but also develop together and are critical for the evolution of the TF ecosystem. This coopetition illuminates how the identified organization collaborates regarding their strategic importance in framework software development projects.

We also analyzed the organizations' collaboration network. While Google dominates the network, various TF releases include numerous organizations. Another remarkable aspect is that the top-influencing organizations contribute more than others to the source code. Therefore, these top organizations play a critical role in developing TF software. However, we identified a slight trend that the influence of the top-contributing organizations decreases with increasing diversity. We also observed a low degree of homophily in code cooperation providing advantages in terms of access, low barriers, and external resources for the entire TF ecosystem.

These results help better understand software developers' and organizations' code-collaborative patterns in OSS ecosystems' evolution. The methodical SNA approach also visualizes human and organizational activities in software development. The findings of this work are essential and deepen the understanding of ecosystem evolution for practitioners and academics in the software engineering discipline. We also established clear theoretical references to essential theories in SNA and software engineering.

Further research is needed to bolster and advance our TF ecosystem study findings. Moreover, further case-study research is needed to explore open coopetition in greater depth and its implications for OSS ecosystems.

REFERENCES

Abufouda, M., & Abukwaik, H. (2017). *On using network science in mining developers collaboration in software engineering: A systematic literature review*. https://arxiv.org/pdf/1712.0086510.5121/ijdkp.2017.7601

Amrit, C. A., Daneva, M., & Damian, D. (2014). Human factors in software development: On its underlying theories and the value of learning from related disciplines. A guest editorial introduction to the special issue. *Information and Software Technology*, *56*(12), 1537–1542. doi:10.1016/j.infsof.2014.07.006

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, *286*(5439), 509–512. doi:10.1126/science.286.5439.509 PMID:10521342

Barbosa, O., & Alves, C. (2011). A systematic mapping study on software ecosystems. *Proceedings of the Workshop on Software Ecosystems 2011*.

Basili, V. R., Shull, F., & Lanubile, F. (1999). Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, *25*(4), 456–473. doi:10.1109/32.799939

Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An Open Source Software for Exploring and Manipulating Networks. *Proceedings of the International AAAI Conference on Web and Social Media*, *3*(1), 361-362. doi:10.1609/icwsm.v3i1.13937

Behfar, S. K., Turkina, E., & Burger-Helmchen, T. (2018). Knowledge management in OSS communities: Relationship between dense and sparse network structures. *International Journal of Information Management*, *38*(1), 167–174. doi:10.1016/j.ijinfomgt.2017.09.004

Bengtsson, M., & Kock, S. (2000). Coopetition" in business networks—To cooperate and compete simultaneously. *Industrial Marketing Management*, *29*(5), 411–426. doi:10.1016/S0019-8501(99)00067-X

Bengtsson, M., & Kock, S. (2014). Coopetition—Quo vadis? Past accomplishments and future challenges. *Industrial Marketing Management*, *43*(2), 180–188. doi:10.1016/j.indmarman.2014.02.015

Biçer, S., Bener, A. B., & Çağlayan, B. (2011). Defect prediction using social network analysis on issue repositories. *Proceedings of the 2011 International Conference on Software and Systems Process,* 63–71. doi:10.1145/1987875.1987888

Burt, R. S. (2009). *Structural holes: The social structure of competition*. Harvard University Press.

Çaglayan, B., & Bener, A. B. (2016). Effect of developer collaboration activity on software quality in two large scale projects. *Journal of Systems and Software*, *118*, 288–296. doi:10.1016/j.jss.2016.03.055

Capiluppi, A., Stol, K.-J., & Boldyreff, C. (2012). Exploring the Role of Commercial Stakeholders in Open Source Software Evolution. In I. Hammouda (Ed.), *IFIP advances in information and communication technology, Open source systems long-term sustainability: 8th IFIP WG 2.13 7International Conference, OSS 2012, Hammamet, Tunisia, September 10 - 13, 2012; Proceedings* (pp. 178–200). Springer. doi:10.1007/978-3-642-33442-9_12

Charette, R. N. (2005). Why software fails. *IEEE Spectrum*, *42*(9), 42–49. doi:10.1109/MSPEC.2005.1502528

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, *81*(6), 961–971. doi:10.1016/j.jss.2007.08.020

Concas, G., Marchesi, M., Monni, C., Orrù, M., & Tonelli, R. (2017). Software quality and community structure in java software networks. *International Journal of Software Engineering and Knowledge Engineering*, *27*(07), 1063–1096. doi:10.1142/S0218194017500401

Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2017). A systematic mapping study of software development with GitHub. *IEEE Access: Practical Innovations, Open Solutions*, *5*, 7173–7192. doi:10.1109/ACCESS.2017.2682323

Dagnino, G. B., & Rocco, E. (2009). *Coopetition strategy: Theory, experiments and cases*. Routledge. doi:10.4324/9780203874301

Dinghofer, K., & Hartung, F. (2020). Analysis of criteria for the selection of machine learning frameworks. *2020 International Conference on Computing, Networking and Communications (ICNC)*, 373–377. doi:10.1109/ICNC47757.2020.9049650

El Asri, I., Kerzazi, N., Benhiba, L., & Janati, M. (2017). From periphery to core: A temporal analysis of Github contributors' collaboration network. In L. M. Camarinha-Matos & H. Afsarmanesh (Eds.), *IFIP advances in information and communication technology, Collaboration in a data-rich world: 18th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2017, Vicenza, Italy, September 18-20, 2017 Proceedings* (pp. 217–229). Springer. doi:10.1007/978-3-319-65151-4_21

Fischbach, K., Gloor, P. A., & Schoder, D. (2009). Analysis of informal communication networks–a case study. *Business & Information Systems Engineering*, *1*(2), 140–149. doi:10.1007/s12599-008-0018-z

Frischbier, S., Gesmann, M., Mayer, D., Roth, A., & Webel, C. (2012). Emergence as competitive advantage - engineering tomorrow's enterprise software systems. In L. Maciaszek (Ed.), *Proceedings of the 14th International Conference on Enterprise Information Systems*: Wrocław, Poland, 28 June - 1 July, 2012 (pp. 181–186). SCITEPRESS. doi:10.5220/0003970501810186

Gerber, A., Molefe, O., & van der Merwe, A. (2010). Documenting open source migration processes for re-use. *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, 75-95. doi:10.1145/1899503.1899512

Ghafele, R., & Gibert, B. (2014). Open growth. *International Journal of Open Source Software and Processes*, *5*(1), 16–49. doi:10.4018/ijossp.2014010102

Ghobadi, S., & D'Ambra, J. (2012). Coopetitive relationships in cross-functional software development teams: How to model and measure? *Journal of Systems and Software*, *85*(5), 1096–1104. doi:10.1016/j.jss.2011.12.027

Gonzalez-Barahona, J. M., & Robles, G. (2013). Trends in free, libre, open source software communities: from volunteers to companies [Aktuelle trends in free-, libre-, und open-source-software-gemeinschaften: Von freiwilligen zu unternehmen]. *Itit*, *55*(5), 173–180. doi:10.1515/itit.2013.1012

Guendouz, M., Amine, A., & Hamou, R. M. (2015). Recommending relevant open source projects on github using a collaborative-filtering technique. *International Journal of Open Source Software and Processes*, *6*(1), 1–16. doi:10.4018/IJOSSP.2015010101

Han, J., Shihab, E., Wan, Z., Deng, S., & Xia, X. (2020). What do programmers discuss about deep learning frameworks. *Empirical Software Engineering*, *25*(4), 2694–2747. doi:10.1007/s10664-020-09819-6

Herbold, S., Amirfallah, A., Trautsch, F., & Grabowski, J. (2021). A systematic mapping study of developer social network research. *Journal of Systems and Software*, *171*, 110802. doi:10.1016/j.jss.2020.110802

Hinds, D., & Lee, R. M. (2009). Communication network characteristics of open source communities. *International Journal of Open Source Software and Processes*, *1*(4), 26–48. doi:10.4018/jossp.2009100102

Holmstrom, H., Conchuir, E., Agerfalk, P., & Fitzgerald, B. (2006). Global software development challenges: a case study on temporal, geographical and socio-cultural distance. *International Conference on Global Software Engineering, 2006*, 3–11. doi:10.1109/ICGSE.2006.261210

Howison, J., Wiggins, A., & Crowston, K. (2011). Validity issues in the use of social network analysis with digital trace data. *Journal of the Association for Information Systems*, *12*(12), 2. doi:10.17705/1jais.00282

Jansen, S., Brinkkemper, S., & Finkelstein, A. (2009). Business network management as a survival strategy: A tale of two software ecosystems. *Proceedings of the first International Workshop on Software Ecosystems 2009*, 34-48.

Jansen, S., Cusumano, M. A., & Brinkkemper, S. (2013). *Software ecosystems: Analyzing and managing business networks in the software industry*. Edward Elgar Publishing. doi:10.4337/9781781955635

Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). A sense of community: A research agenda for software ecosystems. *IEEE 31st International Conference on Software Engineering, 2009,* 187–190. doi:10.1109/ICSE-COMPANION.2009.5070978

Jiang, Q., Lee, Y. C., Davis, J. G., & Zomaya, A. Y. (2018, September 11). *Diversity, productivity, and growth of open source developer communities*. https://arxiv.org/pdf/1809.03725

Joblin, M., Apel, S., Hunsen, C., & Mauerer, W. (2017). Classifying developers into core and peripheral: An empirical study on count and network metrics. *IEEE/ACM 39th International Conference on Software Engineering*, 164–174. doi:10.1109/ICSE.2017.23

Kappelhoff, P. (1987). *Cliquenanalyse. Die Bestimmung von intern verbundenen Teilgruppen in Netzwerken. Techniken Der Empirischen Sozialforschung–Methoden Der Netzwerkanalyse*. Pieper.

Latorre, R., & Suárez, J. (2017). Measuring social networks when forming information system project teams. *Journal of Systems and Software*, *134*, 304–323. doi:10.1016/j.jss.2017.09.019

Lee, J., Lee, H., & Park, J. (2013). Exploring the impact of leadership competencies on team social capital and performance in IT service team. In R. H. Sprague (Ed.), *46th Hawaii International Conference on System Sciences (HICSS), 2013: 7 - 10 Jan. 2013, Wailea, Maui, Hawaii; proceedings* (pp. 4344–4353). IEEE. doi:10.1109/HICSS.2013.224

Linåker, J., Rempel, P., Regnell, B., & Mäder, P. (2016). How firms adapt and interact in open source ecosystems: Analyzing stakeholder influence and collaboration patterns. In M. Daneva (Ed.), Lecture Notes in Computer Science. *Requirements engineering foundation for software quality: 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14-17, 2016 Proceedings* (*Vol. 9619*, pp. 63–81). Springer. doi:10.1007/978-3-319-30282-9_5

Madey, G., Freeh, V., & Tynan, R. (2002). The open source software development phenomenon: An analysis based on social network theory. *Proceedings of the Eighth Americas Conference on Information Systems*, 1806-1813.

Manikas, K., & Hansen, K. M. (2013). Software ecosystems–A systematic literature review. *Journal of Systems and Software*, *86*(5), 1294–1306. doi:10.1016/j.jss.2012.12.026

McPherson, M., Smith-Lovin, L., & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, *27*(1), 415–444. https://www.jstor.org/stable/2678628. doi:10.1146/annurev.soc.27.1.415

Meneely, A., & Williams, L. (2011). Socio-technical developer networks: Should we trust our measurements? *2011 33rd International Conference on Software Engineering (ICSE)*. doi:10.1145/1985793.1985832

Mens, T., & Grosjean, P. (2015). The ecology of software ecosystems. *Computer*, *48*(10), 85–87. doi:10.1109/MC.2015.298

Morgan, L., & Finnegan, P. (2014). Beyond free software: An exploration of the business value of strategic open source. *The Journal of Strategic Information Systems*, *23*(3), 226–238. doi:10.1016/j.jsis.2014.07.001

Nguyen Duc, A., Cruzes, D. S., Hanssen, G. K., Snarby, T., & Abrahamsson, P. (2017). Coopetition of software firms in open source software ecosystems. In A. Ojala & H. H. Olsson (Eds.), *Lecture notes in business information processing, Software business: 8th International Conference* (pp. 146–160). Springer. doi:10.1007/978-3-319-69191-6_10

Oliveira, E., Conte, T., Cristo, M., & Valentim, N. (2018). Influence factors in software productivity — A tertiary literature review. *International Journal of Software Engineering and Knowledge Engineering, 28*(11-12), 1795–1810. 10.1142/S0218194018400296

Ouriques, R. A. B., Wnuk, K., Gorschek, T., & Svensson, R. B. (2019). Knowledge management strategies and processes in agile software development: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, *29*(03), 345–380. doi:10.1142/S0218194019500153

Pryke, S., & Smyth, H. (2012). *The management of complex projects: A relationship approach*. John Wiley & Sons.

Rashid, E., & Prakash, M. (2022). An empirical analysis of inferences from commit, fork, and branch rates of top GitHub projects. *International Journal of Open Source Software and Processes*, *13*(1), 1–16. doi:10.4018/IJOSSP.300751

Runeson, P. (Ed.). (2012). *Case study research in software engineering: Guidelines and examples* (1st ed.). Wiley. doi:10.1002/9781118181034

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. doi:10.1007/s10664-008-9102-8

Schenk, M. (1995). *Soziale Netzwerke und Massenmedien: Untersuchungen zum Einfluss der persönlichen Kommunikation*. Mohr Siebeck.

Schreiber, R. R., & Zylka, M. P. (2020). Social network analysis in software development projects: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, *30*(03), 321–362. doi:10.1142/S021819402050014X

Scott, J. (2013). *Social network analysis* (3rd ed.). SAGE.

Scott, J., & Carrington, P. J. (2011). *The SAGE handbook of social network analysis*. Sage., doi:10.4135/9781446294413

Seker, A., Diri, B., Arslan, H., & Amasyalı, M. F. (2022). Open source software development challenges. In I. R. M. Association (Ed.), *Research anthology on agile software, software development, and testing* (pp. 2134–2164). IGI Global. doi:10.4018/978-1-6684-3702-5.ch102

Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, *52*(7), 1000–1014. doi:10.1287/mnsc.1060.0553

Silveira, K. K., & Prikladnicki, R. (2019). A systematic mapping study of diversity in software engineering: A perspective from the agile methodologies. *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE),* 7-10. doi:10.1109/CHASE.2019.00010

Singh, P. V., Tan, Y., & Mookerjee, V. (2011). Network effects: The influence of structural capital on open source project success. *Management Information Systems Quarterly*, *35*(4), 813–829. doi:10.2307/41409962

Šmite, D., Moe, N. B., Šāblis, A., & Wohlin, C. (2017). Software teams and their knowledge networks in large-scale software development. *Information and Software Technology*, *86*, 71–86. doi:10.1016/j.infsof.2017.01.003

Squire, M. (2014). Forge++: The changing landscape of FLOSS development. *2014 47th Hawaii International Conference on System Sciences*, 3266–3275. doi:10.1109/HICSS.2014.405

Teixeira, J., & Lin, T. (2014). Collaboration in the open-source arena: The webkit case. *Proceedings of the 52nd ACM conference on Computers and people research*, 121–129. doi:10.1145/2599990.2600009

The State of the Octoverse. (2021, February 3). *The State of the Octoverse*. https://octoverse.github.com/

Tidström, A. (2009). Causes of conflict in intercompetitor cooperation. *Journal of Business and Industrial Marketing*, *24*(7), 506–518. doi:10.1108/08858620910986749

Toral, S. L., Martínez-Torres, M. R., & Barrero, F. J. (2009). Virtual communities as a resource for the development of OSS projects: The case of Linux ports to embedded processors. *Behaviour & Information Technology*, *28*(5), 405–419. doi:10.1080/01449290903121394

Uzzi, B. (1997). Social structure and competition in interfirm networks: The paradox of embeddedness. *Administrative Science Quarterly*, *42*(1), 35–67. doi:10.2307/2393808

Vasilescu, B., Posnett, D., Ray, B., van den Brand, M. G., Serebrenik, A., Devanbu, P., & Filkov, V. (2015). Gender and tenure diversity in GitHub teams. In J. Kim (Ed.), *Proceedings of the 33rd Annual CHI Conference on Human Factors in Computing Systems*, 3789–3798. doi:10.1145/2702123.2702549

Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge University Press. doi:10.1017/CBO9780511815478

Wiseman, Y. (2022). Autonomous vehicles. In I. R. M. Association (Ed.), *Research anthology on cross-disciplinary designs and applications of automation* (pp. 878–889). IGI Global. doi:10.4018/978-1-6684-3694-3.ch043

Wohlin, C. (Ed.). (2012). *Experimentation in software engineering*. Springer. doi:10.1007/978-3-642-29044-2

Wohlin, C., Šmite, D., & Moe, N. B. (2015). A general theory of software engineering: Balancing human, social and organizational capitals. *Journal of Systems and Software*, *109*, 229–242. doi:10.1016/j.jss.2015.08.009

Wu, J., & Tang, Q. (2007). Analysis of survival of open source projects: A social network perspective. *PACIS 2007 - 11th Pacific Asia Conference on Information Systems.* https://aisel.aisnet.org/pacis2007/19

Xia, H., Dawande, M., & Mookerjee, V. (2016). Optimal coordination in distributed software development. *Production and Operations Management*, *25*(1), 56–76. doi:10.1111/poms.12408

Yilmaz, M., O'Connor, R. V., & Clarke, P. (2016). Effective social productivity measurements during software development—An empirical study. *International Journal of Software Engineering and Knowledge Engineering*, *26*(03), 457–490. doi:10.1142/S0218194016500194

Yin, R. K. (2014). *Case study research: Design and methods* (5th ed.). Sage. https://worldcatlibraries.org/wcpa/oclc/835951262

Yu, Y., Wang, H., Yin, G., & Wang, T. (2016). Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment? *Information and Software Technology*, *74*, 204–218. doi:10.1016/j.infsof.2016.01.004

Zhou, M., Chen, Q., Mockus, A., & Wu, F. (2017). On the scalability of Linux kernel maintainers' work. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (pp. 27–37). ACM. doi:10.1145/3106237.3106287