Social Network Analysis of Open Source Software: A Review and Categorisation

Kelvin McClean, Des Greer, Anna Jurek-Loughrey

School of Electronics, Electrical Engineering and Computer Science
Queen's University, Belfast BT71NN, United Kingdom

Abstract

Context: As companies have become large users of Open Source Software, it is important that they feel comfortable in their Open Source strategies. One of the critical differences between Open Source and Proprietary Software is the communication networks.
Objective: This paper tries to set a base for understanding how open source teams are structured and how they change. This is vital to understanding Open Source Software Communities.
Method: The paper looks into previous research on Social Network Analysis of Open Source Software, using a systematic literature review. Papers were gathered from Scopus, IEEEXplore and ACM Digital Library, and used or discarded based on predetermined inclusion and exclusion criteria. Research which focuses on the success factors of Open Source Software through Network Analysis is also examined.
Results: A subjective categorisation is established for the papers: Structure, Lifecycle and Communication. It was found that the structure of a project has a large bearing on project success, with developers having previously worked together being indicative of project success. Other structure indicators of success are having a small but structured hierarchy, a diverse user and developer base, and project prominence. However, it was found that information on how these structures appear and evolve over time is lacking, and future research into temporal data models to determine project success information is suggested.
Conclusions: A categorisation of existing research on Social Network Analysis is provided as a basis for further research. Further work into the lifecycle of OSS projects through Social Network Analysis of temporal project information is suggested.

Keywords: Review, Open Source Software, Social Network Analysis

1 Introduction

1.1 Social Networks and Open Source Software

In the past few decades, companies and individuals have increasingly been adopting Open Source Software (OSS) for commercial use, and are taking part in its development at an accelerated pace [1, 2]. In a report detailing the use of OSS, *Red Hat* found that 68% of companies reported an increase in OSS usage in the year of 2018 [3]. A large portion of modern computing is now built on open source; from Apache Web Servers to modern programming languages [4].

An important factor which differentiates OSS from proprietary software is the social component. Developers and users can see the source code, and many may feel motivated to contribute to projects they find useful [5]. Developers will also develop projects for numerous other reasons, such as prestige, entertainment, and the possibility of job opportunities [5].

By contributing to OSS, developers are creating a large amount of data on how they interact with each other, and how they develop software together. Using continued activity as a metric for success, Van Antwerp and Madey [6, 7] have shown that the structure of an OSS network can impact the success of a project; namely that the more developers have worked on a team before, the more likely it is to succeed. Wang [8] also shows the importance of social factors, finding that the larger and more active a project's developer and external network, the more likely a project is to survive. Evidently, this social network data should be considered in future studies on the survival and success of OSS. It can be analysed through Social Network Analysis (SNA), which can provide an overview for how network structures form for certain projects, or for the network as a whole [9].

The necessary dataset for SNA is a set of connections between people [9]. To gather this data for OSS, commit history can be analysed. If people work on the same project at the same time, they might be said to be working together [7]. Similarly, if people comment on the same issue threads, a link between them can be inferred [10]. Modern OSS development sites have additional social network features such as 'following', or 'liking', which can be investigated as well [11].

In this paper, previous research into OSS SNA has been examined. A primary goal of this study is to find classifications under which OSS SNA can be examined. This will allow for a more structured and easily understood summary of past work. The work will then be examined under these categories, and future work will be suggested. It is important to understand what drives OSS, and Social Networks help set OSS apart from proprietary software. By examining SNA of OSS, it will help us understand why OSS grows into the forms it does, and how best companies and individuals may best influence it. Furthermore, a better knowledge of the social attributes of OSS assists in early decision making about the possible composition of a software system, based on judgements about its current usage and support levels [12].

1.2 Aims of this review

With the growing relevance of OSS, it is important that both companies and individuals are aware of how best to influence its development. This review aims to gather all relevant current research into the Social Network effects of OSS. This should allow for examination of the current research network in the field on how to best direct a project to facilitate its success.

Previous research will also be categorised to allow for easier discussion. Through categorising the papers, attention will be drawn to the areas of the field where further research is necessary. After discussion of the papers, these gaps, and any potential future work will be discussed.

2 Methodology

This study follows the guidelines set by Kitchenham [13], which calls for the review to be conducted in three distinct phases:

1. Planning the Review; consisting of identifying the need for a review, and development of a review protocol.
2. Conducting the Review; consisting of identification of research, selection of primary studies, study quality assessment, data extraction & monitoring, and data synthesis.
3. Reporting the Review, a single process phase.

2.1 Planning the Review

Following Kitchenham's guide, planning the review involves creating a "review protocol" which will be followed when carrying out the review. In order to create this protocol, research goals are proposed for this study. To carry out these research goals, the search criteria and procedure are given, along with the inclusion and exclusion criteria.

Research Questions

In order to better understand how open source teams are structured and how they change over time, this work sets out to examine the current research into OSS communities using SNA. The Research Questions (RQs) for this study are summarised in Table 1.

Table 1: List of Research Questions

Research Questions		Motivation
RQ1	How can studies on OSS Networks be categorised?	Social Networks may be easier to understand if they are examined from different perspectives. The papers found in this study will be examined, and divided into relevant categories.
RQ2	How has the past work examined OSS communities from the perspectives of the found categorisations?	Papers will be divided into categories through RQ1. Examining papers using these categories will allow for a more intuitive discussion of the content of the papers.
RQ3	Which attributes of OSS communities' networks can be best used to examine the health and success of a project?	Understanding which factors cause project success or failure can help developers avoid wasted effort in creating a successful project.
RQ4	What gaps are there in the current research?	Having knowledge of the gaps in the existing research will inform what contributions can be made to the OSS and research communities.

These research questions are justified by the need for a basis to summarise and discuss current work on SNAs. The authors are unaware of any previous literature review of SNA of OSS. In the case of RQ1, a published classification will facilitate research in and discussion of SNA for OSS and provide a reference structure for future work. In answering RQ2 the usefulness of this categorisation is demonstrated and at the same time an insight is developed into current and past efforts to understand SNA in OSS. Answering RQ3 is of particular interest to practitioners who may be considering whether to use OSS products. Given the often large investment required for software systems, knowing which projects are likely to survive and/or become stable would be very useful to the community. Equally, avoiding products from communities with Social Networks not conducive to long term survival or at high risk of failure could avoid significant loss. Finally in RQ4, knowing about existing efforts and existing knowledge will provide a resource to inspire and possibly direct future research.

In trying to answer RQ1, an initial literature search was carried out on categorisation of OSS networks but this did not yield results. After a preliminary study of papers relating to SNA of OSS it is proposed that the following categories are potentially useful:

1. Structure, which will include papers in which the form of the network is examined;
2. Lifecycle, which will include papers in which the temporal data and evolution of the project is examined; and
3. Communication, which will include papers in which communication is analysed on a fine-grain detail; for example, a paper may look at sentiment in email logs.

The intention here is to use these categories allow discussion of existing research and inform on similarities and differences in previous work, thus being useful in answering RQ2. These categories are derived from an informal investigation of the area and are not necessarily independent. For example, a paper may look at the temporal data of the structure of the network. Similarly, a

Table 2: List of Search Criteria

Base Search	Permutations
Open Source Software	Open Source Software, OSS, FLOSS, Free Open Source Software, FLOSS
Community	Community, Communities, Communit*
Social Network Analysis	Social Network Analysis, SNA, Social Network*

paper may not necessarily look at any of these categories specifically. The categories were chosen due to the nature of OSS development where there is a lot of data that changes rapidly. Utilising temporal measures and fine-grain record analysis can yield more information than base structural analysis. By using these categories, this information can be given more attention.

In addressing RQ3, these papers will then be combined, and information on how the structure of a network, its lifecycle, and the communications of individuals can contribute to the success or survival of OSS will be presented. This will allow for an assessment of previous research, and provides value as the success of a project most likely does not derive from any one of the categories.

Using the results of RQ3, RQ4 asks about gaps in the current research. The categorisation in RQ2 will allow some future work to be suggested.

Search Criteria

The base search criteria for this study were Open Source Software, Community, and Social Network Analysis. These criteria were put through a number of permutations, which can be found in Table 2. Papers were also limited by release date to being released between 2010 and the date of the search, March 25th 2019.

These search strings will be searched for in the Abstract and Title of a paper.

2.2 Conducting the Review

Search Procedure

Using the Search Criteria, papers were searched for on Scopus, ACM Digital Library, and IEEEXplore. Only empirical papers are included, in the field of Computer Science. This returned a total of 314 papers.

Inclusion & Exclusion Criteria

To ensure that papers are relevant, they are included and discarded based on Inclusion and Exclusion criteria. Inclusion and exclusion criteria are shown on Table 3. These Inclusion Criteria (IC) are the standards a paper must meet to be considered for inclusion in the review. Similarly, the Exclusion Criteria (EC) are the standards by which a paper will be rejected from the review. They were designed to be as broad as possible while also discarding papers that did not focus on using

Inclusion Criteria	Exclusion Criteria
Papers must be empirical	Papers must not only describe the use of OSS tools for SNA of other networks
Papers must be focused on OSS SNA	Papers must not be focused on areas other than OSS
Papers must be primary studies	Papers must not look at suggesting bugs, reviewers, projects, etc. to OSS developers
	Papers must not be merely opinion-based

Table 3: Inclusion & Exclusion Criteria

SNA on OSS. Papers were required to be empirical so that their research questions can be clearly verified or rejected, and the results of the study are not open to interpretation or based mainly on opinion. Many papers used SNA to suggest projects, bugs, etc. to OSS developers, which did not give information on the development of OSS networks, so those papers were discarded.

Duplicates were discarded, leaving 285 papers. Abstracts were read, and papers which did not meet the IC, or met the EC were discarded, leaving 54 papers. Finally, papers were read in full, and eight papers which did not meet the IC or met the EC were also discarded: four because they were opinion-based/not empirically based; two because they related only to the use of OSS tools in doing SNA; one that was not focused on SNA for OSS; and one which was a secondary study not relating to OSS. This left 46 papers.

Data Extraction and Synthesis

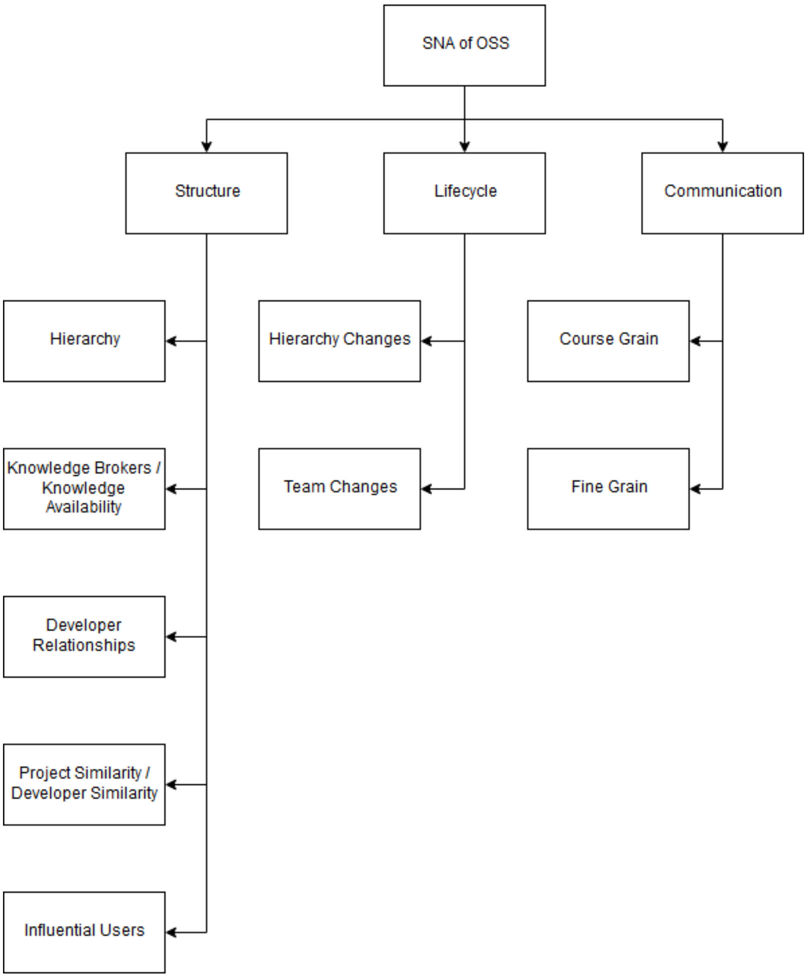

Figure 1: Categories and Subcategories of Papers

The remaining papers were examined, and it was found that they could be studied under the context of Structure, Lifecycle and Communication. Examining papers through the perspective of Structure will allow us to examine how the past work has categorised the organisation of OSS teams, Lifecycle will allow us to look in closer detail at how these structures change, and Communication will examine the relationships that can be found through SNA. The papers were examined again three times, once from the context of each of the categories. This divided the papers again into sub-categories, which were the trends and topics covered many times throughout the papers. These can be seen in Figure 1, the subcategories arising during the process of investigating RQ1. At this stage, the contents of the papers were also summarised for report. Factors for papers involving success were also collected in this section.

The papers, and the categories which they are divided into, can be found in Table 4.

Name	**Structure**	**Lifecycle**	**Communication**
A genetic search of patterns of behaviour in OSS communities	X		
Adoption of Free Libre Open Source Software (FLOSS): A Risk Management Perspective			
Analysis of activity in open-source communities using social network analysis techniques	X		X
Analysis of the core team role in open source communities	X		
Analysis of virtual communities supporting OSS projects using social network analysis	X		
Analyzing Leadership Dynamics in Distributed Group Communication	X	X	X
Analyzing the social ties and structure of contributors in Open Source Software community	X		
Applying Centrality Measures to the Behavior Analysis of Developers in Open Source Software Community	X		
Automatically Modeling Developer Programming Ability and Interest Across Software Communities			
Boundary Spanners in Open Source Software Development: A Study of Python Email Archives	X		X
Coding together in a social network: Collaboration among GitHub users	X		
Communication in open source software development mailing lists			X
CPDScorer: Modeling and evaluating developer programming ability across software communities			
Drawing the big picture: Temporal visualization of dynamic collaboration graphs of OSS software forks		X	X
Empirical Study on the Evolution of Developer Social Networks	X	X	
Evolution of open source software networks	X	X	
Exploring decision-making processes in python			X
Exploring the patterns of social behavior in Github	X		
Expressions of Sentiments during Code Reviews: Male vs. Female			X
Formal in the informal: A multi-level analysis of core python developers' tweets			X
From periphery to core: A temporal analysis of github contributors' collaboration network	X	X	
How Microblogging Networks Affect Project Success of Open Source Software Development	X		X
Hyper contextual software security management for open source software			
Investigating and projecting population structures in open source software projects: A case study of projects in GitHub	X	X	
Lessons learned from applying social network analysis on an industrial Free/Libre/Open Source Software ecosystem	X		
Linchpin developers in open source software projects	X		
Managing knowledge sharing in distributed innovation from the perspective of developers: empirical study of open sourcesoftware projects in China	X		
Mining Developer Mailing List to Predict Software Defects			X
Network analysis of evolving software-systems		X	
Networks, social influence, and the choice among competing innovations: Insights from open source software licenses	X		
On clusters in open source ecosystems Open Source Software developer and project networks	X		
Overcoming challenges in global software development: The role of brokers	X		
Ranking developer candidates by social links	X		
Relating and clustering free/libre open source software projects and developers: A social network perspective	X		X
Reputation management in an open source developer social network: An empirical study on determinants of positive evaluations	X		
Social Activities Rival Patch Submission for Prediction of Developer Initiation in OSS Projects	X		
Social Network Analysis of Developers' and Users' Mailing Lists of Some Free Open Source Software			X
Stochastic actor-oriented modeling for studying homophily and social influence in OSS projects			X
Successful or unsuccessful open source software projects: What is the key?	X		X
The Importance of Social Network Structure in the Open Source Software Developer Community	X		
The meso-level structure of F/OSS collaboration network: Local communities and their innovativeness	X		
The role of Internet in the development of future software projects	X		
The study of open source software collaborative user model based on social network and tag similarity	X		
Understanding a developer social network and its evolution	X		
Using social network analysis to inform management of open source software development	X	X	
Validity of network analyses in Open Source Projects	X		X

Table 4: List of Papers

3 Results

3.1 Datasets

It is important to recognise that a range of data sources has been used in research into SNA of OSS. The sources range from using single projects with a small number of contributors to datasets of thousands of contributors over thousands of projects. Of the 46 papers, 11 involved studies of networks for single projects with a range of sources including the repositories for these projects, email/discussion forums associated with them or bug reports. Studies involving a group of projects of between 2 and 12 projects accounted for 15 of the papers. In these, quite a few study the mailing lists or forums for the projects as well as various metrics on contribution activity. The remaining 20 of the 46 papers were studies of larger datasets often involving extraction of data from a repository host platform such as GitHub, Sourceforge or Ohloh or subset of them. There were also several cases of less widely used repository platforms. In these larger datasets, the focus is often on contributors and contributions data but other data has been used such as license data and data collected directly from developers via questionnaires.

3.2 RQ1: How can studies on OSS Networks be categorised?

During the initial literature search and during the detailed reading of the papers not excluded due to the exclusion/inclusion criteria, a meaningful categorisation emerged in that papers could be grouped through the categories: Structure, Lifecycle, and Communication. There is no claim that this is the best or only categorisation, but we propose that this is useful for further discussion of SNA of OSS.

3.3 RQ2: How has the past work examined OSS communities from the perspectives of the found categorisations?

Examining existing work on SNA of OSS using the categories of Structure, Lifecycle, and Communication provides an organisation with which to discuss previous findings. Papers were studied in detail and the sub-categorisation as shown in Figure 1 was developed. This allows RQ2 to be addressed, looking at papers in detail with respect to these subcategories.

3.3.1 Structure

Following examination of the past work, the discussion on structure of OSS Community Networks is naturally broken into the categories of Hierarchy, Knowledge Brokers & Knowledge Availability, Risk, Developer Relationships, and Project Similarity.

Hierarchy

OSS development is usually initiated either by a single developer, such as Linus Torvalds for the Linux project, or by a core team of developers, such as in the development of Python [14]. As development continues, new developers join the project, some of whom may not be as strongly connected to the project as the initial developers. These developers are known as periphery developers. This two-tiered hierarchy is the most common in Open Source Development, and has become known as the core-periphery model [15, 16].

Martínez-Torres looking at 12 virtual communities for Linux Debian ports [14] found that there were four stages to a project's hierarchical development:

1. Small communities which have high centralisation and are single tiered.
2. Small centralised networks which contain a two-tier system of project-leads and developers.
3. Projects with high local connectivity which are larger and contain many layers of developers. These projects tend to have the lowest number of "free-loaders".
4. Large communities in which a high number of interactions occur and there is a clear hierarchy.

Projects typically begin in the first stage, with new developers being brought in onto the lower tier of development over time. In an extension of this work Martínez-Torres [17] found that the core team of developers are responsible for bringing in these new developers, showing former connections are valuable. Similarly, Allaho and Lee [18] using very large datasets from Ohloh and GitHub, found that as new developers join, developers with higher levels of expertise will partner with newer developers, adding them into the project. This effect may diminish as time goes on however, with He et al. [19] finding that Sourceforge developers that were new to a project began working together, then picked up contacts within the core network after some time.

There is evidence, based on a sample of activity from four Apache OSS projects, that developers move within a hierarchical structure, with developers making many links to other developers moving into the core team over time, [20]. El Asril et al. [16] using a sample of five projects, looked at how developers would move from core to periphery over time. They found that the move would happen, unless a developer was heavily involved in multiple areas of a project. Aljemabi et al. [21] found that most members of the core team remained there, with some developers moving between periphery and core over the course of the project's development.

Martínez-Torres [22] found that the more centralised the core team of developers, the more successful the project will be while Hong et al. [23] discovered that the overall structure of Developer Networks was more homogeneous than those found in other social networks. The centrality of an individual in a project is how "central" they are to every other developer, that is, how many connections they have, compared to the total number of developers in the project. The centrality of a project takes the centrality of all developers and divides it by the number of developers. Crowston et al. [15] found that as a project neared the end of its development cycle, the project became more decentralised. This was based on events and emails from two Instant Messaging client projects. Unavoidably, inferences are therefore made based on communications between developers as opposed to contributions. More recent work by Aljemabi et al. [21] found that OSS developer communities in fact tend to become more tight-knit and centralised as development continues, while also becoming larger. That work is based on 12 projects on GitHub which the authors state are "famous, mature, and successful" and uses data on watchers, commits, issues, pull requests, and comments to establish weighted links between developers and thus a model of the developer social network. Platforms such as GitHub evidently lend themselves to SNA, since they are modelled in part on creating and building a social network.

There has been some study on the roles played in developer social networks. Onoue et al. [24] looking at a subset of 90 projects from GitHub found a distinction between coding and non-coding contributors, and stated that as time went on, non-coding contributors were more and more likely to get involved in the project. Kumar [25], investigating 96 projects from Sourceforge, derived several metrics which may give critical information to project managers. For example, it was found that the number of clusters apparent in the social network should mirror the number of development tasks at any one time. Similarly, core-developers could serve as a cross-disciplinary knowledge hub in a large project, so the core team should contain members from each coding and non-coding groups.

Knowledge Brokers & Knowledge Availability

An issue with OSS is the distribution of knowledge. Without distinct project managers, it can be a challenge for teams to find developers with certain knowledge sets. Knowledge brokers have naturally arisen to solve this issue. Knowledge brokers are developers who have contacts in multiple communities, and act as ties between them. The role of knowledge brokers is discussed by Manteli et al. [26], who look at brokers in a project perspective. Using a single 2-year research project with 37 members, they found that without brokers, clustering has a negative effect on transactive memory; that is, "the memory which members of a group develop for... retrieving expertise knowledge within their group". Brokers can help mitigate this loss of transactive memory on projects with large amounts of clustering.

Knowledge brokers initially consist of the core developers of a project, and as a project becomes more developed, a knowledge broker will be a developer with a key interest in a specific area of development. When projects form, both Toral et al. [27] and Sharma et al. [28] using single projects found that knowledge brokers also tend to be the team's core team and decision makers. Toral et al. also found that the number of brokers was associated with the centrality of the team; that is, the more connected the entire team was, the less knowledge brokers were necessary. Corona et al. [29] looked at the roles of knowledge brokers in multiple projects and found that they tended to have the same role across any projects they worked on. It was also found that developers tended to be knowledge brokers across similar spaces, and developers working in vastly different spaces are rare. These findings show that the developers who are knowledge brokers are specialised in a specific area of development.

Chen et al. [30], using data from communications in four projects and subsequently using questionnaires to 403 developers [31] found that these core developers are less important than previously suggested, instead finding that all developers share knowledge which they gain through working on other projects. However, mediation of this knowledge sharing process through central figures was shown to increase team motivation. Kavaler and Filkov [32] agreed with Chen et al.'s model based on two case studies, and found that knowledge moves through all developers in "knowledge circles", as developers move in different groups of a project.

Conaldi et al. [33] researched the formation of knowledge brokers in the larger OSS developer network. It was found that a large portion of the community was connected, and brokers connected highly centralised clusters together. This overall network supports the theory of knowledge brokers distributing knowledge throughout OSS ecosystems as a whole. Beyond that, there is little previous work on how knowledge is effectively curated and shared, especially in larger projects highly distributed contributors from diverse domains.

Developer Relationships

Through working together, developers form bonds with the fellow workers. These relationships often form due to reciprocity; developers are likely to help developers who have helped them in the past, as found by Celinska [34] in a study of GitHub users, or just generally worked with before, as found by He et al. [19] and Van Antwerp [35] in older studies based on Sourceforge projects. Another factor which may influence which developers form relationships is homophily; if a developer has more in common with another developer, they are more likely to work together in the future [32]. Teixeira et al. [36] discussed whether researchers from the same firm worked with each other. It was found that while companies will often push for certain releases, hired developers often do not work with any co-workers, and in fact co-operate heavily with industry competitors. However, it seems that the smaller a project, the easier relationships form from it, with Syed et al. [37] finding that developers who work in larger connected clusters do not report as tight friendships with other developers. These relationships are shown to have advantages in development. Van Antwerp [7] found that projects containing these repeated links are more likely to be successful. Hu et al. [38] discovered that developers would be more likely to give each other positive evaluations when they've worked together positively in the past, but especially so if they were from the same area and had similar programming ability and interests.

Project Similarity & Developer Similarity

Several studies have researched how projects and developers can be clustered. This is valuable as developers are more likely to work with developers with whom they share interests [39, 32]. Huang et al. [40] gathered static project information on features such as project language, target audience, project space, packages used, license, etc. and used them as features in a clustering algorithm. They find that clusters of projects could be found by grouping smaller sections of these features together. They combined this cluster data with developer cluster data to attempt to find OSS ideologists.

Suggesting relevant projects to developers is also a key way in which new members can be recruited. By analysing project tag similarity, weighted by collaborations between developers, Chen et al. [41] suggested a model for project similarity. This was used to find projects that are in similar spaces to one another, and suggest future projects for developers to work on.

Influential Users

As OSS development often lacks the official team structures found in traditional software development, looking at how users influence each other is important, due to the effect it can have on the path of development. Exemplifying this, Yu et al. [42] found that the more followed a user was on GitHub, a project they worked on was more likely to be taken up by other developers. Similarly, Yang et al. [43] using a dataset from Ohloh found that as developers gained more followers, they asserted a higher influence on their projects, and the rate at which they gained followers increased. However, Yu et al. [42] also found that not all users working on a project follow users they are directly working with, instead using the feature to keep track of influential users outside of their working circle. Therefore, they conclude that followers do not directly lead to influence in one particular field.

Singh and Phelps [44] found that when choosing a license, the more core members had used a license before, the more likely it would be chosen for a new project. However, a project leader with experience in multiple other teams would be more likely to resist influence from other developers.

3.3.2 Lifecycle

Hierarchy Changes

Looking at how the hierarchy of a project changes over time can help in understanding how a project develops.

El Asri et al. [16] looked at the differences between developers who stayed in core, and those who would move to periphery. They found that developers who remained in the core team made many commits, with those commits being split across multiple files. It was found that developers who focused on one area of the project no longer had a reason to remain a core member of the team when that area was completed. While the research of Aljemabi and Wang [21] also using

GitHub data does not contradict this, it is found that following some changes at the inception of the project, the core team only changed in small amounts, with only between 3 and 12 percent of the team changing between periphery and core.

As well as the changing core-periphery structure of the team, the overall structure can change over time as well. Babic and Grbac [45] and Hong et al. [23] examined the evolution of OSS network structure for a given product, and found that developers become less separated and less clustered over time. Crowston et al.'s [15] research had earlier stated that there was a trend over time for a more distributed leadership hierarchy. Their graphs also show spikes in centrality as leadership changes from one core developer to another. Azarbakht and Jensen [46] worked on visualising major events in project evolution. In a centrality graph, they found that developers who are about to leave a project lost project centrality in the months before leaving the project. Onoue et al. [24] examined population structures of committing and non-committing developers over time. They found that whether developers begin by committing or not depends heavily on project, but in general, the longer a developer remains part of a project, the higher the chance that they will become a committing developer. This transition usually happens within the first three months of a developer being involved in a project.

Team Changes

Aljemabi and Wang [21] looked at how developer networks change over time, and found that as the project matures, the networks become more connected, and less clustered. They also become harder to join, with fewer people joining mature projects. Van Antwerp [35, 7] looked at repeated connections in Sourceforge, and found that developers were more likely to work on a project with a developer they had worked with before, and that projects with large amounts of these repeated connections remained active for longer than those without. Part of this may be explained by a link between satisfaction with a project and a feeling of connection. Azarbakht and Jensen [46] modelled project forks through collaboration graphs based on records of communications between developers in three projects, and found that dissatisfied users would become less involved with other users, with a visible split in the team after a fork is made.

3.3.3 Communication

During the review an observation was made that papers dealing with Communication can broken down into two categories: Coarse-Grain papers, which gather metrics from a broad look at communication methods, and fine-grain, which analyse the content of the communication methods in detail.

Course-Grain

Mailing lists along with other forms of communication can be analysed to determine information about project structure. Coarse-Grain examination looks at past research that has examined the networks created by these forms of communication. Communication involvement could be used as a proxy measure of developer importance. For example, Martínez-Torres [14] looked at email logs for 12 Debian Linux ports, and built a network graph from who is part of which email chains. These graphs were then used to model developers' changing positions in the project; the more communication a developer takes place in, the more likely they were to be a member of the core team. This backs up the findings of Crowston et al. [15] who earlier made a similar contribution using email chains for two Instant Messaging products, using involvement in email logs as a measure of developer importance. More recently While Sharma et al. [28] did not use raw developer involvement as a measure of importance, they use communication graphs derived from several Python mailing lists to find users who are active in multiple areas. They created network graphs from email chains, and suggested that developers most active in multiple areas are likely to be acting as knowledge brokers for the project, thus also inferring importance.

Mailing lists are also used by external developers who are showing an interest in the project's development. Xuan et al. [20], as well as Gharehyazie et al. [47] examined communication behaviours of developers in a group of Apache projects prior to committing to a project. They found that developers who remain active after joining a project tend to get involved in the mailing lists prior to committing. Further they found that the amount of communication between them and other developers is an very strong indicator of later patch acceptance.

Azarbakht and Jensen [46] looked at communication break-down, and found that before breaking into a fork of a project, dissatisfied developers will lower the number of people who they remain in contact with, effectively creating a small cluster to themselves. Keertipati et al. [48] examined how decisions are made in mailing list communities, and found that there are three general models: Benevolent Dictator, Rotating Dictator, and Meritocracy. These models look at decision making threads, and find that either all decisions are presided by one individual, a series of changing individuals, or are made based on the area of expertise of the task and developer.

However, mailing lists may not be an accurate representation of any decision making process. This is argued by Guzzi et al. [49], who looked at the formal team leaders, and found that roughly 75% of communications happen without the presence of this core team present in the mailing chain.

Fine-Grain

Fine-Grain communication looks at how the contents of messages can be examined. Paul et al. looked at sentiment expression differences between men and women in code reviews [50]. They found that women are less likely to display sentiment, and that men are more likely to express negative sentiment against women developers' submissions, and more likely to withhold positive comments. Zhang et al. [51] used sentiment analysis to detect defect discussion in mailing lists, and found that when negative sentiment is shown, there is a high chance that there is discussion on defects in those threads.

Yasir et al. [52] researched the content in Python developers tweets. They found that around 45% of core developers' tweets are Python related, and around 2% are decision oriented. It was also discovered that developers tweet mainly in the area relevant to their own work, which indicated developers were mainly discussing their own work, rather than the Python project as a whole.

3.4 RQ3: Which attributes of OSS communities' networks can be best used to examine the health and success of a project?

To answer Research Question 3, the papers from all categories which analyse success factors of OSS will be analysed. Van Antwerp [35, 7] found that they key to continued activity in projects is the presence of repeated connections. In projects where past connections existed, the average defined "activity percentile" was 78, compared to 70 on projects where no such links existed.

Martínez-Torres et al. [53] found that a successful project exhibits a strong hierarchical structure. In later research, Martínez-Torres [14] complemented this by showing that projects which contain more developers show higher levels of hierarchy. However, this research also showed that the most active projects are of a moderate size, with high levels of local connectivity, and small levels of hierarchy between the local layers.

Chen et al. [30] using four cases from the APDPlat platform found that a knowledge-sharing user-base and developer-base was a key to a successful project. This base provides what the researchers call “distributed innovation”. This shared knowledge is based on User Knowledge, presented through comments and emails, as well as Developer Knowledge, which derives from Participation Motivation, Social Network, and Organisational Culture. The theory presented by Chen et al. suggests that this shared knowledge will help the development team overcome any challenges which may arise in its development cycle. In a similar vein, Yang et al. [43] found that discussion on a project will affect its success, based on data related to the Ohloh platform. Their research also supported a rich-get-richer model, where larger projects attract more attention.

3.5 RQ4: What gaps are there in the current research?

The categorisation that emerged in this study has proven useful for grouping findings. Of course, other categorisations may be just as useful and we make no claim that this is the final or best way to study existing literature on SNA of OSS. It is naturally subjective and there are numerous ways of answering RQ1. Therefore, there is scope for either developing the current categorisation or presenting an alternative.

In looking at work on the structure of SNA of OSS we discovered that this was a popular and fruitful area of research. The idea of a hierarchical structure has been widely accepted e.g. [21] and the drift to centrality well-researched. There has been limited research though on movement between projects and this might produce interesting results.

The role of knowledge brokers is also interesting with recent work [31, 32] showing that mediation in this respect proves useful. Thus, there would be huge benefit in looking at how knowledge can be curated, promoted and managed via social networks. In a similar domain, previous work has pointed out that developers tend to work with like-minded developers but there has been some work that suggests this is not as significant as expected, making the link to increased use of technology over face-to-face interactions [36]. Much of the work to date relates to single project data or to platforms before the popularity of platforms such as GitHub that explicitly encourage the building of social networks via platform features such as “following” or “starring”. There is further opportunity to relate these new features to social network analysis. This clearly would be worthy of further investigation.

After examining the list of papers per category in Table 4, it can be concluded that there is an opportunity for further research into how temporal project information, and especially how that information impacts project success. Only eight of the 54 papers addressed temporal data or evolution of projects. Of particular note, an observation can be made about the lack of reporting on how knowledge is shared in a distributed OSS team and the role of the social network in this. All of these address matters to do with the evolution of the structure of the network or the role of developers over time, particularly hierarchy changes. Missing from this is extensive work on how the quality of code is affected over time by the social network structure. There may therefore be scope to investigate the changes in quality over time and if the structure of the team has any effect on this. One aspect of this might be to monitor quality attributes over time using existing statically measured design quality and technical debt metrics [54]. Changes in team membership over time is also underrepresented in recent work, especially the tendency of projects to be forked [46]. Analysing how projects change over time could give greater insight into the project lifecycle of OSS. For this reason, future work should be directed into analysing OSS network’s evolution from a temporal perspective. One potentially interesting gap is in looking for clusters present in various projects, the change of knowledge brokers, the strength of the core-periphery project hierarchy, and of centrality as a whole. the relationship of network structure evolution with code quality, measured through static code analysis might also be of value.

Of crucial concern to organisations considering the use of an OSS product is whether a product has stability and whether it has a long term future. This might be hard to predict and very little work has been published on how the extent that networks are connected has on project success or project health. The papers which answered RQ3 mainly used static structural information to analyse successful projects. Indeed there is no agreed definition of success for OSS projects but this and other proxy measures can be used, where available. There is an opportunity here for future research in this area and for correlation of success or usage with social network metrics.

4 Conclusions

In this paper, previous research into the SNA of OSS Communities has been examined and categorised. Past research has been divided into the categories of Structure, Lifecycle, and Communication. This has emerged in response to RQ1: "How can studies on OSS Networks be categorised?" These categories were chosen through a natural analysis of the papers, and represent a structured way to discuss papers on the topic of SNA of OSS. The further breakdown of these categories as shown in Figure 1, was arrived at iteratively during the review process. As papers were read and analysed for their the respective categories, the subcategories emerged and were gradually refined, as shown.

Following this categorisation, papers were examined through those categories, answering RQ2: "How has the past work examined OSS communities from the perspectives of the found categorisations?" Analysing the Structural papers, it was found that the core-periphery structure of projects is a key sign to a strong project, and the existence of core and periphery developers is well established. The hierarchy of a project, both in terms of centrality models and the core-periphery model, is well established, as is knowledge broker theory. Other aspects of project structure, such as the study of developer relationships and influential users are less well fleshed out, with little research on how those structures impact the development of the project.

Knowledge Brokers transfer knowledge both internally in projects, and between projects, although Chen et al. [30] and Kavaler and Filkov's [32] research suggest that knowledge brokers are not singular individuals, but collective knowledge is brought from every developer. Developers were found more likely to help those they had prior links with, especially those who had helped with their work previously. As users gained more followers, they were also found to gain more influence in the direction of projects they are involved in.

Looking at papers which deal with Lifecycle analysis, it was found that generally, projects' leadership does not change large amounts over time, although as a project nears the end of its development cycle, it tends to become less hierarchical with a more evenly distributed leadership. Projects as a whole become tighter knit as the development cycle continues, with teams becoming tighter knit over time. When examining project lifecycle, research has shown that the core team of a project generally remains in place, with minor fluctuations, although the project gets more distributed over time. Communication through mailing lists is analysed by several researchers, who use it to model developer importance and networks, and discover that developers looking to join a team will be more likely to be accepted based on previous email communications.

The past research into developer communication through SNA has found that the amount of communication before a developer commits to a project is a valuable metric for determining their commitment to it. This research also shows that sentiment analysis is a tool that can be utilised to determine factors about developer behaviour.

We also sought to establish an answer to RQ3: "Which attributes of OSS communities' networks can be best used to examine the health and success of a project?" Success is defined through a number of various metrics, with continued activity being the most common. Previous research finds that repeated connections, a strong hierarchical structure, a diverse user and developer base, and discussion activity are used to model the success of a project. To summarise the past research on project success:

- Repeated links lead to higher activity.
- A small but structured hierarchy is most effective, with centralised teams for different areas of the project.
- Shared knowledge of a diverse user and developer base will bring varied knowledge to the project.
- Discussion of projects by others will bring users into the project.
- Large projects will generally attract more attention.

Finally, we looked for possibilities for future areas of research in SNA of OSS in answering RQ4: "What gaps are there in the current research?" Using the information from the previous research questions, future work is suggested into how the temporal lifecycle information of a project may influence its success. Further research into using temporal data models to determine project success information was suggested. Researching this could give greater insight into the project lifecycle of OSS.

As OSS is becoming part of our everyday lives, it is important that its formation is better understood. To help realise this, this study collects previous research in the field and proposes to categorise it in a useful way, in the process summarising existing work and offering a basis for continuing this work. Future research can build upon this knowledge, and help us understand how OSS is developed, and the communities which develop it, and should focus on the development of OSS communities. Focusing on creating links between relationship development over time and project success could provide an interesting perspective on OSS evolution.

References

[1] L. F. Dias, I. Steinmacher, and G. Pinto, "Who drives company-owned oss projects: internal or external members?," *Journal of the Brazilian Computer Society*, vol. 24, p. 16, Dec 2018.

[2] I. Ahmed, D. Forrest, and C. Jensen, "A case study of motivations for corporate contribution to FOSS," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2017* (R. P., H. A.Z., and S. A., eds.), vol. 2017-Octob, (School of EECS, Oregon State University, Corvallis, OR, United States), pp. 223–231, IEEE Computer Society, 2017.

[3] J. R. H. Whitehurst, "The State of Enterprise Open Source," tech. rep., Red Hat, 2019.

[4] S. Androutsellis-Theotokis, D. Spinellis, M. Kechagia, G. Gousios, *et al.*, "Open source software: A survey from 10,000 feet," *Foundations and Trends® in Technology, Information and Operations Management*, vol. 4, no. 3–4, pp. 187–347, 2011.

[5] J. A. Roberts, I.-H. Hann, and S. A. Slaughter, "Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects," *Management Science*, vol. 52, no. 7, pp. 984–999, 2006.

[6] M. Van Antwerp, "Evolution of open source software networks," in *OSS 2010 Doctoral Consortium, Collocated with the 6th International Conference on Open Source Systems, OSS 2010*, (University of Notre Dame, United States), pp. 25–39, 2010.

[7] M. Van Antwerp and G. Madey, "The importance of social network structure in the open source software developer community," *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 1–10, 2010.

[8] J. Wang, "Survival factors for Free Open Source Software projects: A multi-stage perspective," *European Management Journal*, vol. 30, no. 4, pp. 352–371, 2012.

[9] E. Otte and R. Rousseau, "Social network analysis: a powerful strategy, also for the information sciences," *Journal of Information Science*, vol. 28, no. 6, pp. 441–453, 2002.

[10] J. Wu, K. Goh, and Q. Tang, "Investigating success of open source software projects: A Social network perspective," *Proceedings of the Twenty Eighth International Conference on Information Systems (ICIS 2007), Montreal (QC), Canada, 9-12 December, 2007*, p. Paper 105, 2007.

[11] H. Borges and M. Tulio Valente, "What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 2018.

[12] D. Greer and R. Conradi, "Software project initiation and planning–an empirical study," *IET software*, vol. 3, no. 5, pp. 356–368, 2009.

[13] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[14] M. R. Martínez-Torres, "A genetic search of patterns of behaviour in OSS communities," *Expert Systems with Applications*, vol. 39, no. 18, pp. 13182–13192, 2012.

[15] K. Crowston, A. Wiggins, and J. Howison, "Analyzing leadership dynamics in distributed group communication," *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 1–10, 2010.

[16] M. El Asri, I., Kerzazi, N., Benhiba, L., Janati, "From periphery to core: A temporal analysis of github contributors' collaboration network," *18th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2017 Vicenza, Italy, September 18–20, 2017 Proceedings*, 2017.

[17] M. d. R. Martínez-Torres, "Analysis of activity in open-source communities using social network analysis techniques," *Asian Journal of Technology Innovation*, vol. 22, no. 1, pp. 114–130, 2014.

[18] M. Y. Allaho and W.-C. Lee, "Analyzing the social ties and structure of contributors in open source software community," *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pp. 56–60, 2014.

[19] P. He, B. Li, and Y. Huang, "Applying centrality measures to the behavior analysis of developers in open source software community," *Proceedings - 2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications, CGC/SCA 2012*, pp. 418–423, 2012.

[20] Q. I. Xuan, C. Fu, and L. I. Yu, "Ranking Developer Candidates By Social Links," *Advances in Complex Systems*, vol. 17, no. 7-8, 2015.

[21] M. A. Aljemabi and Z. Wang, "Empirical study on the evolution of developer social networks," *IEEE Access*, vol. 6, pp. 1–1, 2018.

[22] M. R. Torres, S. L. Toral, M. Perales, and F. Barrero, "Analysis of the core team role in open source communities," *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2011*, pp. 109–114, 2011.

[23] Q. Hong, S. Kim, S. C. Cheung, and C. Bird, "Understanding a developer social network and its evolution," *IEEE International Conference on Software Maintenance, ICSM*, vol. 1, no. 612108, pp. 323–332, 2011.

[24] S. Onoue, H. Hata, A. Monden, and K. Matsumoto, "Investigating and projecting population structures in open source software projects: A case study of projects in GitHub," *IEICE Transactions on Information and Systems*, vol. E99D, no. 5, pp. 1304–1315, 2016.

[25] S. Kumar, "Using social network analysis to inform management of open source software development," *Proceedings of the Annual Hawaii International Conference on System Sciences*, vol. 2015-March, pp. 5154–5163, 2015.

[26] C. Manteli, B. Van Den Hooff, H. Van Vliet, and W. Van Duinkerken, "Overcoming challenges in global software development: The role of brokers," *Proceedings - International Conference on Research Challenges in Information Science*, pp. 1–9, 2014.

[27] S. L. Toral, M. R. Martínez-Torres, and F. Barrero, "Analysis of virtual communities supporting OSS projects using social network analysis," *Information and Software Technology*, vol. 52, no. 3, pp. 296–303, 2010.

[28] P. N. Sharma, B. T. R. Savarimuthu, and N. Stanger, "Boundary Spanners in Open Source Software Development: A Study of Python Email Archives," in *24th Asia-Pacific Software Engineering Conference, APSEC 2017* (L. J., Z. H., L. X., and H. M., eds.), vol. 2017-Decem, (Department of Information Science, University of Otago, Dunedin, New Zealand), pp. 308–317, IEEE Computer Society, 2018.

[29] E. I. M. Corona and B. Rossi, "Linchpin Developers in Open Source Software Projects," in *12th IASTED International Conference on Software Engineering, SE 2013*, (Free University of Bozen-Bolzano, 39100 Bolzano, Italy), pp. 701–708, 2013.

[30] X. Chen, D. Probert, Y. Zhou, and J. Su, "Successful or unsuccessful open source software projects: What is the key?," *Proceedings of the 2015 Science and Information Conference, SAI 2015*, vol. 1, no. 34, pp. 277–282, 2015.

[31] X. Chen, Y. Zhou, D. Probert, and J. Su, "Managing knowledge sharing in distributed innovation from the perspective of developers: empirical study of open source software projects in China," *Technology Analysis and Strategic Management*, vol. 29, no. 1, pp. 1–22, 2017.

[32] D. Kavaler and V. Filkov, "Stochastic actor-oriented modeling for studying homophily and social influence in OSS projects," *Empirical Software Engineering*, vol. 22, no. 1, pp. 407–435, 2017.

[33] G. Conaldi and F. Rullani, "The meso-level structure of F/OSS collaboration network: Local communities and their innovativeness," *IFIP Advances in Information and Communication Technology*, vol. 319 AICT, pp. 42–52, 2010.

[34] D. Celińska, "Coding together in a social network: Collaboration among GitHub users," in *9th International Conference on Social Media and Society, SMSociety 2018*, (University of Warsaw, Faculty of Economic Sciences, Warsaw, Poland), pp. 31–40, Association for Computing Machinery, 2018.

[35] M. Van Antwerp, "Evolution of open source software networks," in *OSS 2010 Doctoral Consortium, Collocated with the 6th International Conference on Open Source Systems, OSS 2010*, (University of Notre Dame, United States), pp. 25–39, 2010.

[36] J. Teixeira, G. Robles, and J. M. González-Barahona, "Lessons learned from applying social network analysis on an industrial Free/Libre/Open Source Software ecosystem," *Journal of Internet Services and Applications*, vol. 6, no. 1, 2015.

[37] S. Syed and S. Jansen, "On clusters in open source ecosystems," *CEUR Workshop Proceedings*, vol. 987, pp. 13–25, 2013.

[38] D. Hu, J. L. Zhao, and J. Cheng, "Reputation management in an open source developer social network: An empirical study on determinants of positive evaluations," *Decision Support Systems*, vol. 53, no. 3, pp. 526–533, 2012.

[39] S. Daniel, V. Midha, A. Bhattacherhjee, and S. P. Singh, "Sourcing knowledge in open source software projects: The impacts of internal and external social capital on project success," *Journal of Strategic Information Systems*, vol. 27, no. 3, pp. 237–256, 2018.

[40] K. Y. Huang and N. Choi, "Relating and clustering free/libre open source software projects and developers: A social network perspective," *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 1–10, 2011.

[41] X. Chen and Y.-h. Pan, "The Study of Open Source Software Collaborative User Model Based on Social Network and Tag Similarity.," *Journal of Electronic Commerce Research*, vol. 15, no. 1, pp. 77–86, 2014.

[42] Y. Yu, G. Yin, H. Wang, and T. Wang, "Exploring the patterns of social behavior in GitHub," in *1st International Workshop on Crowd-Based Software Development Methods and Technologies, CrowdSoft 2014*, (National Laboratory for Parallel and Distributed Processing, School of Computer Science, National University of Defense Technology, Changsha, 410073, China), pp. 31–36, Association for Computing Machinery, Inc, 2014.

[43] X. Yang, D. Hu, and D. M. Robert, "How microblogging networks affect project success of open source software development," *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 3178–3186, 2013.

[44] P. Singh and C. Phelps, "Networks, Social Influence, and the Choice Among Competing Innovations: Insights from Open Source Software Licenses," *Information Systems Research*, vol. 24, no. November 2014, pp. 539–560, 2012.

[45] S. G. Babic and T. G. Grbac, "Network analysis of evolving software-systems," *2017 25th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2017*, 2017.

[46] A. Azarbakht and C. Jensen, "Drawing the Big Picture: Temporal Visualization of Dynamic Collaboration Graphs of OSS Software Forks," *IFIP Advances in Information and Communication Technology*, vol. 427, pp. 41–50, 2014.

[47] M. Gharehyazie, D. Posnett, and V. Filkov, "Social activities rival patch submission for prediction of developer initiation in OSS projects," *IEEE International Conference on Software Maintenance, ICSM*, pp. 340–349, 2013.

[48] S. Keertipati, S. A. Licorish, and B. T. R. Savarimuthu, "Exploring decision-making processes in Python," in *20th International Conference on Evaluation and Assessment in Software Engineering*, pp. 1–10, 2016.

[49] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. Van Deursen, "Communication in open source software development mailing lists," in *10th International Working Conference on Mining Software Repositories, MSR 2013*, (Department of Software and Computer Technology, Delft University of Technology, Netherlands), pp. 277–286, 2013.

[50] R. Paul, A. Bosu, and K. Z. Sultana, "Expressions of Sentiments During Code Reviews: Male vs. Female," *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 26–37, 2018.

[51] Y. Zhang, B. Shen, and Y. Chen, "Mining developer mailing list to predict software defects," in *Proceedings - Asia-Pacific Software Engineering Conference, APSEC* (G. Y-G., K. G., and C. S., eds.), vol. 1, (School of Software, Shanghai JiaoTong University, Shanghai, 200240, China), pp. 383–390, IEEE Computer Society, 2014.

[52] M. Yasir, K. Michael, B. T. R. Savarimuthu, and S. A. Licorish, "Formal in the informal: A multi-level analysis of core python developers' tweets," *Proceedings - 25th Australasian Software Engineering Conference, ASWEC 2018*, pp. 151–160, 2018.

[53] M. R. Martínez-Torres, S. L. Toral, F. Barrero, and F. Cortés, "The role of Internet in the development of future software projects," *Internet Research*, vol. 20, no. 1, pp. 72–86, 2010.

[54] M. Mohan, D. Greer, and P. McMullan, "Technical debt reduction using search based automated refactoring," *Journal of Systems and Software*, 2016.