Longitudinal Analysis of Collaboration Graphs of Forked Open Source Software Development Projects

Amirhosein “Emerson” Azarbakht
School of Electrical Engineering and Computer Science
Oregon State University
azarbaka@oregonstate.edu

Dissertation Draft

Work Under Supervision of Prof. Carlos Jensen
Ph.D. Committee Members:
Prof. Alex Groce
Prof. Drew Gerkey
Prof. Ron Metoyer
Prof. A. Morrie Craig

March 13, 2017

Contents

List of Figures

List of Tables

1 Abstract

Social interactions are a ubiquitous part of our lives, and the creation of online social communities has been a natural extension of this phenomena. Free and Open Source Software (FOSS) development efforts are prime examples of how communities can be leveraged in software development, where groups are formed around communities of interest, and depend on continued interest and involvement.

Forking in FOSS, either as an non-friendly split or a friendly divide, affects the community. Such effects have been studied, shedding light on how forking happens. However, most existing research on forking is post-hoc. In this study, we focus on the seldom-studied run-up to forking events. We propose using statistical modeling of longitudinal social collaboration graphs of software developers to study the evolution and social dynamics of FOSS communities. We aim to identify measures for influence and the shift of influence, measures associated with unhealthy group dynamics, for example a simmering conflict, in addition to early indicators of major events in the lifespan of a community.

We use an actor-oriented approach to statistically model the changes a FOSS community goes through in the run-up to a fork. The model represents the tie formation, breakage, and maintenance. It uses several (more than two, up to 10) snapshots of the network as observed data to estimate the influence of several statistical effects on formation of the observed networks. Exact calculation of the model is not trivial, so, instead we simulate the changes and estimate the model using a Markov Chain Monte Carlo approach.

When we find a well-fitting model, we can test our hypothesis about model parameters, the contributing effects using T-tests and Multivariate Analysis of Variance Between Multiple Groups (Multivariate ANOVA). Our method enables us to make meaningful statements about whether the network dynamics depends on particular parameters/effects with a p-value, indicating the statistical significance level.

This approach may help predict formation of unhealthy dynamics, which is the first step toward a model that gives the community a heads-up when they can still take action to ensure the sustainability of the project.

2 Introduction

Social networks are a ubiquitous part of our social lives, and the creation of online social communities has been a natural extension of this phenomena. Social media plays an important role in software engineering, as software developers use them to communicate, learn, collaborate and coordinate with others [55]. Free and Open Source Software (FOSS) development efforts are prime examples of how community can be leveraged in software development, where groups are formed around communities of interest, and depend on continued interest and involvement to stay alive [38].

Community splits in free and open source software development are referred to as forks, and are relatively common. Robles et al. [46] define forking as "when a part of a development community (or a third party not related to the project) starts a completely independent line of development based on the source code basis of the project."

Although the bulk of collaboration and communication in FOSS communities occurs online and is publicly accessible for researchers, there are still many open questions about the social dynamics in FOSS communities. Projects may go through a metamorphosis when faced with an influx of new developers or the involvement of an outside organization. Conflicts between developers' divergent visions about the future of the project may lead to forking of the project and dilution of the community. Forking, either as an acrimonious split when there is a conflict, or as a friendly divide when new features are experimentally added, affect the community [9].

Previous research on forking ranges from the study by Robles et al. [46] that identified 220 significant FOSS projects that have forked over the past 30 years, and compiled a comprehensive list of the dates and reasons for forking (listed in Table 1, and depicted by frequency in Figure 1), to the study by Baishakhi et al. [7] on post-forking porting of new features or bug fixes from peer projects. It encompasses works of Nyman on developers' opinions about forking [40], developers motivations for performing forks [35], the necessity of code forking as tool for sustainability [39], and Syeed's work on sociotechnical dependencies in the BSD projects family [56].

Most existing research on forking, however, is post-hoc. It looks at the forking events in retrospect and tries to find the outcome of the fork; what happened after the fork happened; what was the cause of forking, and such. The run-up to the forking events are seldom studied. This leaves several questions unanswered: Was it a long-term trend? Was the community polarized, before forking happened? Was there a shift of influence? Did the center of gravity of the community change? What was the tipping point? Was it predictable? Is it ever predictable? We are missing that context.

Additionally, studies of FOSS communities tend to suffer from an important limitation. They treat community as a static structure rather than a dynamic process. Longitudinal studies on open source forking are rare. To better understand and measure the evolution, social dynamics of forked FOSS projects, and integral components to understanding their evolution and direction, we need new and better tools. Before making such new tools, we need to gain a better understanding of the context. With this knowledge and these tools, we could help projects reflect on their actions, and help community leaders make informed decisions about possible changes or interventions. It will also help potential sponsors make informed decisions when investing in a project, and throughout their involvement to ensure a sustainable engagement.

Identification is the first step to rectify an undesirable dynamic before the damage is done. A community that does not manage growing pains may end up stagnating or dissolving. Managing growing pains is especially important in the case of FOSS projects, where near half the project contributors are volunteers [20]. Oh et al. [41] have argued that openness in FOSS is "[...] generally perceived as having a positive connotation, however, the term can also be interpreted as referring to some nonconstructive characteristics, such as unobstructed exit, susceptible, vulnerable, fragile, lacking effective regulation, and so on. The unobstructed exit and lack of regulatory force inherent in the FOSS community can result in a community's susceptibility and vulnerability to herded exits by its participants. Commercial vendor intervention, an alternative project becoming available, and licensing issues can result in some original core members ceasing to provide their loyal service for the community, which can prompt their coworkers to leave as well" [41]. Identification of recipes for success or stagnation, sustainability or fragmentation may lead to a set of best practices and pitfalls.

I propose to use temporal social network analysis to study the evolution and social dynamics of FOSS communities. Specifically, we propose using a longitudinal exponential family random graph statistical model to investigate the driving forces in formation and dissolution of communities. Additionally, to complement the statistical study, we propose doing a qualitative interview study for validating the findings. With these techniques we aim to identify better measures for influence, shifts of influence, measures associated with unhealthy group dynamics, for example a simmering conflict, in addition to early indicators of major events in the lifespan of a community. One set of dynamics we are especially interested in, are those that lead FOSS projects to fork.

Table 1: The main reasons for forking as classified by Robles and Gonzalez-Barahona [46]

Reason for forking	**Example forks**
Technical (Addition of functionality)	Amarok & Clementine Player
More community-driven development	Asterisk & Callweaver
Differences among developer team	Kamailio & OpenSIPS
Discontinuation of the original project	Apache web server
Commercial strategy forks	LibreOffice & OpenOffice.org
Experimental	GCC & EGCS
Legal issues	X.Org & XFree

Table 2: The frequency of main reasons for forking as classified by Robles and Gonzalez-Barahona [46]

Reason	**Frequency**
Technical	60 (27.3%)
Discontinuation of the original project	44 (20.0%)
More community-driven development	29 (13.2%)
Legal issues	24 (10.9%)
Commercial strategy forks	20 (9.1%)
Differences among developer team	16 (7.3%)
Experimental	5 (2.3%)
Not Found	22 (10.0%)

This proposal report is organized as follows: Section 3 presents related literature on open source social communities, the gap in the literature, and discusses why the issue needs to be studied. Then, Section 4 presents our research objective and research questions. After that, in section 5 presents how data gathering, statistical modeling, and qualitative analysis are proposed to be done, as well as our initial hypotheses. Section 7 presents the work that needs to be done. Section 8, shows the proposed timeline for the research. Lastly, section 9 the threats to validity are discussed.

3 Related Work

The free and open source software development communities have been studied extensively. Researchers have studied the social structure and dynamics of team communications [10][22][26][27][34], identifying knowledge brokers and associated activities [52], project sustainability [34][39], forking [38], requirement satisfation [17], their topology [10], their demographic diversity [30], gender differences in the process of joining them [29], and the role of age and the core team in their communities [2][3][16][58]. Most of these studies have tended to look at community as a static structure rather than a dynamic process [15]. This makes it hard to determine cause and effect, or the exact impact of social changes.

Post-forking porting of new features or bug fixes from peer projects happens among forked projects [7]. A case study of the BSD family (i.e., FreeBSD, OpenBSD, and NetBSD, which evolved from the same code base) found that 10-15% of lines in BSD release patches consist of ported edits, and on average 26-58% of active developers take part in porting per release. Additionally, They found that over 50% of ported changes propagate to other projects within three releases [7]. This shows the amount of redundant work developers need to do to synchronize and keep up with development in parallel projects.

Visual exploration of the collaboration networks in FOSS communities was the focus of a study that aimed to observe how key events in the mobile-device industry affected the WebKit collaboration network over its lifetime. [57] They found that *coopetition* (both competition and collaboration) exists in the open source community; moreover, they observed that the "firms that played a more central role in the WebKit project such as Google, Apple and Samsung were by 2013 the leaders of the mobile-devices industry. Whereas more peripheral firms such as RIM and Nokia lost market-share" [57].

The study of communities has grown in popularity in part thanks to advances in social network analysis. From the earliest works by Zachary [59] to the more recent works of Leskovec et al. [31][32], there is a growing body of quantitative research on online communities. The earliest works on communities was done with a focus on information diffusion in a community [59]. The study by Zachary investigated the fission of a community; the process of communities splitting into two or more parts. They found that fission could be predicted by applying the Ford-Fulkerson min-cut algorithm [19] on the group's communication graph; "the unequal flow of sentiments across the ties" and discriminatory sharing of information lead to subcommunities with more internal stability than the community as a whole.[59]

The dynamic behavior of a network and identifying key events was the aim of a study by Asur et al [1]. They studied three DBLP co-authorship networks and defined the evolution of these networks as following one of these paths: a) Continue, b) k-Merge, c) k-Split, d) Form, or e) Dissolve. They defined four possible transformation events for individual members: 1) Appear, 2) Disappear, 3) Join, and 4) Leave. They compared groups extracted from consecutive snapshots, based on the size and overlap of every pair of groups. Then, they labeled groups with events, and used these identified events [1].

Table 3: The behavioral measures used by Asur et al. [1]

Metrics	**Meaning**
Stability	Tendency of a node to have interactions with the same nodes over time
Sociability	Tendency of a node to have different interactions
Influence	Number of followers a node has on a network and how its actions are copied and/or followed by other nodes. (e.g., when it joins/leaves a conversation, many other nodes join/leave the conversation, too)
Popularity	Number of nodes in a cluster (how crowded a sub-community is)

The communication patterns of free and open source software developers in a bug repository were examined by Howison et al. [26]. They calculated out-degree centrality as their metric. Out-degree centrality measures the proportion of times a node contacted other nodes (outgoing) over how many times it was contacted by other nodes (incoming). They calculated this centrality over time "in 90-day windows, moving the window forward 30 days at a time." They found that "while change at the center of FOSS projects is relatively uncommon," participation across the community is highly skewed, following a power-law distribution, where many participants appear for a short period of time, and a very small number of participants are at the center for long periods. Our proposed approach is similar to theirs in how we form collaboration graphs. Our approach is different in terms of our project selection criteria, the metrics we examine, and our research questions.

Table 4: The measures of diversity used by Kunegis et al. [30]

Network property	**Network is diverse when**	**Diversity Measures**
Paths between nodes	Paths are long	Effective diameter
Degrees of nodes	Degrees are equal	Gini coefficient of the degree distribution
Communities	Communities have similar sizes	Fractional rank of the adjacency matrix
Random walks	Random walks have high probability of return	Weighted spectral distribution
Control of nodes	Nodes are hard to control	Number of driver nodes

The tension between diversity and homogeneity in a community was studied by Kunegis et al. [30]. They defined five network statistics, listed in Table 4, used to examine the evolution of large-scale networks over time. They found that except for the diameter, all other measures of diversity shrunk as the networks matured over their lifespan. Kunegis et al. [30] argued that one possible reason could be that the community structure consolidates as projects mature.

Community dynamics was the focus of a more recent study by Hannemann and Klamma [23] on three open source bioinformatics communities. They measured "age" of users, as starting from their first activity and found survival rates and two indicators for significant changes in the core of the community. They identified a survival rate pattern of 20-40-90%, meaning that only 20% of the newcomers survived after their first year, 40% of the survivors survived through the second year, and 90% of the remaining ones, survived over the next years. As for the change in the core, they suggested that a falling maximum betweenness in combination with an increasing network diameter as an indicator for a significant change in the core, e.g., retirement of a central person in the community. Our initial network-specific study built on their findings, and the evolution of betweenness centralities and network diameters for the projects in our study are explained in the following sections.

4 Research Goals

Social interactions reflect the changes the community goes through, and so, it can be used to describe the context surrounding a forking event. Social interactions in FOSS can happen, for example, in the form of mailing list email correspondence, bug report issue follow-ups, and source code co-authoring.

We consider the following three of the seven main reasons for forking [46] to be socially related: (1) Personal differences among developer team, (2) The need for more community-driven development, and (3) Technical differences for addition of functionality.

Table 5: The socially-related reasons for forking

Reason	**Frequency**
Differences among developer team	16 (7.3%)
More community-driven development	29 (13.2%)
Technical	60 (27.3%)

By socially-related, we mean, the forking categories that should have left traces in the developers' interactions data. Such traces may be identified using longitudinal modeling of the interactions, without digging into the contents of the communications. These three reasons are (1) Personal differences among developer team, (2) The need for more community-driven development, and (3) Technical differences for addition of functionality.

As an example of how these traces of forking can be identified, if a fork occurred because of a desire for "more community-driven development", we should see interaction patterns in the collaboration data showing a strongly-connected core that is hard to penetrate for the rest of the community (i.e. the power stayed in the hands of the same people throughout, as developers joined and left.)

In this study, we plan to analyze, quantify and visualize how the community is structured, how it evolves, and the degree to which community involvement changes over time.

Specifically, our overall research objective is to identify these traces/social patterns associated with different types of undesirable forking?

In the following and in section 5.7, we will discuss our research objectives and research questions in depth.

Do forks leave traces in the collaboration artifacts of open source projects in the period leading up to the fork?
To study the properties of possible social patterns, we need to verify their existence. More specifically, we need to check whether the possible social patterns are manifested in the the collaboration artifacts of open source projects, e.g., mailing list data, issue tracking systems data, source code data. This is going to be accomplished by statistical modeling of developer interactions as explained in more detail in section 5.

Do different types of forks leave different types of traces?
If forks leave traces in the collaboration artifacts, do forks exhibit different social patterns? Are there patterns that exemplify these categories? For example, is there a prototypical "personal differences" fork collaboration pattern? If so, do different forking reasons have distinctly different social patterns associated with them? Is a project labeled as a "technical differences" fork only a "technical differences" fork? Or, alternatively, can they be a mix of several reason categories?

We are going to investigate this by statistical modeling of the interaction graphs, as explained in detail in section 5.3.

What are the key indicators that let us distinguish between different types of forks?
What quantitative measure(s) can be used as an early warning sign of an inflection point (fork)? Are there metrics that can be used to monitor the odds of change, (e.g. forking-related patterns), ahead of time? This will be accomplished by statistical modeling of developer interactions as explained in more detail in section 5.

To validate what our quantitative approach finds, and to account and check for possible confounding factors, we will interview and survey people from the studied forked projects. We will also analyze the sentiments in the content of the messages send and received by the top contributors of the project in the month leading to the forking events will be analyzed.

5 Methodology

Detecting change patterns, requires gathering relevant data, cleaning it, and analyzing it. In the following subsections, we describe the proposed process in detail. Figures 1 and 2 show the overview of the methodology.

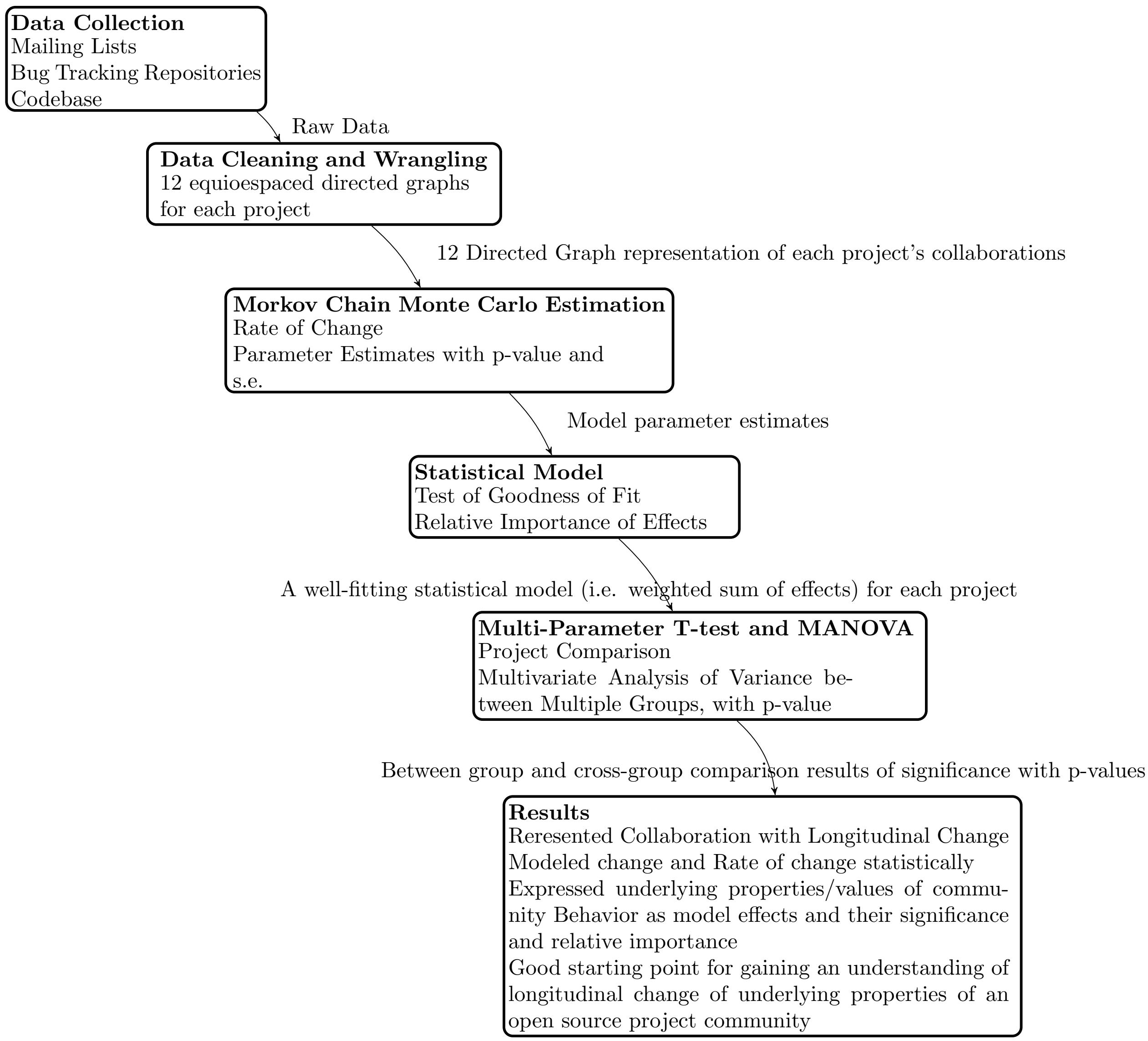


Figure 1: The methodology in a glance

Figure 2: The methodology overview

5.1 Phase 1: Data Collection

5.1.1 The Case of Undesirable forking

To find patterns uniquely associated with undesirable forks, we need to gather data on projects from the following three categories:

Table 6: The three types of projects for which data is collected in this study

Type of forking	**Abbreviation**
Undesirable & socially-related forking	U.F.
Other socially-related forking	H.F.
No forking at all (as the control group)	No.F.

We define Undesirable and socially-related Forking (U.F.) as the projects forked because of *Personal differences among developers team*, or because of the need for *more community-driven development*. These situations are undesirable because they imply an increase in cost of maintenance, redundant or wasted efforts, and a lost shared value. One-fifth (20.5%) of the 220 forked projects fall into this category [46].

Other socially-related (H.F.) is defined as projects forked because of *technical differences (Addition of functionality)*. More than a quarter (27.3%) of the 220 forked projects fall into this category.

These three categories are the socially-related categories, because for forks driven by personal conflicts, the need more community-driven development, and technical differences, we expect to see the social context during the run-up to forking captured by the projects' social artifacts.

To find projects in U.F. and H.F. categories, we looked at the list of all significant open source software forks in the past three decades as compiled by Robles and Gonzalez-Barahona [46]. Their study found the reasons behind each fork, listed in Table 1. We applied three selection criteria to the 220 forked projects on that list to find projects in U.F. and H.F. categories. A project was short-listed as either a U.F. or H.F. if **a)** the forking was recent, i.e., happened after the year 2000, **b)** its data was existent and available to access and download online, or was made accessible to us after our requests; and **c)** the project had a sizable developer community, i.e., more than a dozen developers, which means it would be large enough to make a sociogram for a meaningful statistical analysis. For the No.F. category, we chose well-known,

Table 7: List of projects in U.F. H.F., and No.F. categories selected based on the criteria described in section 5.1 for our study

Projects	**Reason for forking**	**Year forked**	**Type**
Kamailio & OpenSIPS	Differences among developer team	2008	U.F.
ffmpeg & libav	Differences among developer team	2011	U.F.
Asterisk & Callweaver	More community-driven development	2007	U.F.
rdesktop & FreeRDP	More community-driven development	2010	U.F.
freeglut & OpenGLUT	More community-driven development	2004	U.F.
Amarok & Clementine Player	Technical (Addition of functionality)	2010	H.F.
Apache CouchDB & BigCouch	Technical (Addition of functionality)	2010	H.F.
Pidgin & Carrier	Technical (Addition of functionality)	2008	H.F.
MPlayer & MPlayerXP	Technical (Addition of functionality)	2005	H.F.
Ceph	Not forked	-	No.F.
Python	Not forked	-	No.F.
OpenStack Neutron	Not forked	-	No.F.
GlusterFS	Not forked	-	No.F.

stable projects that had been around for a while (two years), and had large communities; and had not forked; and were similar in size of the development team. Similarity in size is a constrain that our statistical method imposes for the results to be meaningful, as described in detail in section 5.3. The preceding criteria resulted in the projects listed in Table 7. The rest of the 220 forked projects were discarded, because they did not meet the described filtering criteria.

5.1.2 Data Sources

The data sources to collect are **a)** developer mailing lists, where developers' interact by sending and receiving emails, **b)** Issue(bug) tracking systems, where developers interact by reporting an issue/bug, discussing how to resolve it, and closing the issues, and , **c)** Source-code repository contribution logs, where developers interact by modifying the code, and/or working on the same source files. The sociograms will be formed based on interactions among developers in any of the preceding data sources.

The time period for which data was collected is one year leading to when the fork happened. This should supposedly capture the social context prior, and at the time of the fork.

5.2 Phase 2: Sociogram Formation and Statistical Study

Social connections and non-connections can be represented as graphs, in which the nodes represent actors (developers) and the edges represent the interaction(s) between actors or lack thereof. Such graphs can be a snapshot of a network – a static sociogram – or a changing network, also called a dynamic sociogram. In this phase, we process interactions data to form a communication sociogram of the community.

Two types of analysis can be done on sociograms: Either a *cross-sectional* study, in which only one snapshot of the network is looked at and analyzed; or a *longitudinal* study, in which several consecutive snapshots of the network are looked at and studied. We are interested in patterns in the run-up to forks, therefore, unlike most existing research on forking, we do a longitudinal study.

A longitudinal study can look at the sociograms in the following two distinct approaches:

1. A *network-specific* approach, which can be called a skin-deep *measurement-specific* look, in which, we focus on the observed networks, measure their properties, to describe the structure of the *observed networks* with numeric summaries/descriptors, e.g., in Figures 12 and 13. Many studies measuring centralities as their only metrics, fall into this category.

2. A *population-processes* approach, in which we treat an observed network as one instance from a set of all possible networks with the same number of nodes, and with similar characteristics. In a population-processes approach, the observed network is only good to help us understand the social forces/processes that generated it, because we are interested in the forces/processes that underlie the structure and changes of the network; these reflect the values shared or not shared by the community members, and represent the group behavior, and behavior change trends that lead to a forking event.

Figures 12 and 13 are good examples of the expressiveness and limitations of a network-specific approach. Figure 12 shows the normalized change in the number of active nodes (developers) and edges (interaction between developers) for eight FOSS projects from U.F. and H.F. categories.

The normalized number of nodes and edges for the U.F. forks (i.e., second and third row) show a sudden sharp decrease around the month of fork. This can be because of dissolution of the development team, who either left the project, or stopped being active contributors as much as before. Figure 13 shows normalized change of diameter over time for the same projects. It is hard to make sense of normalized diameter changes across projects, and this demonstrates the limitations of network-specific approaches, as they may, or may not (often the case), help us understand the network dynamics.

As an example of expressiveness, figure 12 shows that a measurements-only approach shows us the trends of change or no-change in the measured metrics. It also shows one limitation of such an approach: even though we can see trend changes, it is hard to make sense of such trends given the short scope of the measurements. Additionally, it is hard to constitutes a reference point and back it up mathematically or statistically.

The following subsection lists the reasons why we need to use a statistical model, used in the *population-processes* approach.

5.2.1 Why a statistical model is needed?

One may wonder why we should look beyond the observed network. We can do measurements on the observed networks data, and get some descriptive statistics. This would be a superficial look at the data, and even though necessary, is not good enough and has several shortcomings. There is another way of looking at these networks, in a less-superficial way; namely, finding a model that fits the data and its longitudinal change. So, instead of a superficial look at the data, we can find a well-fitting statistical model of our observed interactions network. The following lists the reasons why we need to go beyond a superficial measurement-only approach.

1. For the *observed network*, a small observation error or sampling error, and the uncertainty involved with real-world communication, can result in large perturbation in the numeric descriptors used to describe static graphs.

2. The traditional *network-specific* approach assumes edges in sociograms are statistically independent, (and/or identically distributed). This can be misleading, as, social network data are relational. For example, in real-life human communication, the likelihood of forming ties with friends of a friend is higher than a stranger, as Balance Theory suggests [24].

3. In population-specific approach, we try to identify the social forces that have formed the observed network, by simulation a population of similar networks of the same characteristics. After finding the statistical distribution of network population, then we can compare the observed network to the population distribution of possible networks of that size, and see how significant and likely it is to observe such a graph, as compared to observing a randomly-generated graph. This is useful, because it gives us a reference point to compare our observed graph with, and to weed out the properties generated by random processes, and to find the statistically significant network statistics. In short, with a statistical model, we can draw inferences about whether certain network structures and substructures are more commonly observed in the observed network than might be expected by chance [45].

4. Stochastic models capture the regularities in the processes that caused the network ties form, as well as variability that are hard to model otherwise. A model that considers stochasticity allows us to understand the uncertainty associated with an observed network. It makes it possible to learn about the distribution of possible networks for a given specification of a model [45].

5. Different social processes may manifest similar network structures. For example, clustering in a network might be because of structural effects, e.g., structural balance, or through node-level effects, e.g., homophily. To determine which one is the case in our observed network, a statistical model that incorporates both covariates can help. We then can assess the contribution of each covariate, and infer which social process underlies the observed network [45].

6. Localized processes might not scale to the entire network well. The combination of the overall structure and the localized processes is hard

to investigate without a model. (This micro-macro difference may be investigated through model simulation.)[45]

In summary, a measurement-only approach would not be able to explain the longitudinal changes in an open source community's network properly. It can confuse us, and it can mislead us. Our initial study, described in appendix section 2.2, is an example of a measurement-only approach, which shows such limitations. This initial study guided in the proper direction; namely, trying to find a statistical model that can explain the networks' longitudinal changes. In the following section 5.3, we described such a model.

5.3 The Statistical Model

Longitudinal evolution of a network data is the result of many small atomic changes occurring between the consecutively observed networks. In our case, software developers are the actors in the networks, and they can form a connection with another developer, break off an existing connection, or maintain their status quo. These are the four possibilities of atomic change within our evolving networks: (1) forming a new tie; (2) breaking off an existing tie; (3) maintaining a non-connection; and (4) maintaining a connection. We assume a continuous-time network evolution, even though our observations are made at two or more discrete time points.

The state-of-the-art in studying longitudinal social networks, is the idea of *actor-oriented models* [51], based on a model of developers changing their outgoing ties as a consequence of a stochastic optimization of an *objective function*. This framework assumes that the observed networks at discrete times, are outcomes of a continuous-time Markov process. In the case of open source developers, the actor-oriented model, can be informally described as OpenSourceDeveloper-oriented model, in which, it is assumed that developers are in charge of their communication and collaboration choices. They choose to have interactions with certain other developers and/or they choose to stop having interactions with another developer. In short, they have autonomy in choosing their connections.

Let the data for our statistical developer-oriented model be M repeated observations on a network with g developers. The M observed networks (at least two) are represented as directed graphs with adjacency matrices $X(t_m) = (X_{ij}(t_m))$ for $m = 1, ..., M$, where i and j range from a to g. The variable X_{ij} shows whether at time t there exists a tie from i to j (value 1) or not (value 0). Be definition, $\forall i, X_{ii} = 0$ (i.e. the diagonal of the adjacency matrices).

In order to model the network evolution from $X(t_1)$ to $X(t_2)$, and so on, it is natural to treat the network dynamics as the result of a series of small atomic changes, and not bound to the observation moment, but rather as a more of less continuous process. In this way, the current network structure is a determinant of the likelihood of the changes that might happen next [14].

For each change, the model focuses on the developer whose tie is changing. We assume that developer i has control over the set of outgoing tie variables $(X_{i1}, ..., X_{ig})$ (i.e. the i^{th} row of the adjacency matrix). The network changes one tie at a time. We call such an atomic change a *ministep*. The moment at which developer i changes one of his ties, and the kind of change that he makes, can depend on attributes represented by observed covariates, and the network structure. The moment is stochastically determined by the *rate function*, and the particular change to make, is determined by the *objective function* and the *gratification function*. We cannot calculate this complex model exactly. Rather than calculating exactly, we estimate it using a Monte Carlo Markov Chain method. The estimated model is used to test hypotheses about the forked FOSS communities.

These above three functions and their definitions taken from [50] are explained in detail the following subsections.

5.3.1 Rate Function

The *rate function* $\lambda_i(x)$ for developer i is the rate at which developer i's outgoing connections changes occur. It models how frequently the developers make ministeps.
The rate function is formally defined [50] by

$$\lambda_i(x) = \lim_{dt \to 0} \frac{1}{dt} \mathrm{P}\big(X_{ij}(t+dt) \neq X_{ij}(t) \quad for \quad some \quad j \in \{i, ..., g\} | X(t) = x\big). \tag{1}$$

The simplest specification of the rate of change is that all developers have the same rate of change of their ties.

5.3.2 Objective Function

The *objective function* $f_i(s)$ for developer i is the value attached to the network configuration x.

The idea is that, given the opportunity to make a change in his outgoing tie variables $(X_{i1}, ..., X_{ig})$, developer i selects the change that gives the greatest increase in the objective function. We assume that if there is difference between developers in their objective functions, these differences can be represented based on the model covariates [50].

Let's denote the present network by $x = X(t)$. The new network that would be the result of a ministep (i.e. changing a single tie variable x_{ij} into its opposite $1 - x_{ij}$) by developer i is denoted by $x(i \mapsto j)$. This choice is modeled in the following way:

Let $U(j)$ denote a random variable which represents the attraction for i to j. We assume that these U_j are random variables distributed symmetrically about 0, and independently generated for each new ministep. In this way, the developer i chooses to change his tie variable with the other developer j for whom the following value [50] is the highest:

$$f_i(x(i \mapsto j)) + U(j) \tag{2}$$

This is a short-sighted stochastic optimization rule: short-sighted because only the situation one step after the current step is considered; and stochastic because the unexplained part is modeled by a random variable $U(j)$.

$U(j)$ is chosen to have Gumbel distribution with mean 0 and scale parameter 1. In this way, the probability that developer i chooses to change x_{ij} for any particular j, given that such a change happens, is [50],

$$p_{ij}(x) = \frac{exp(f_i(i \mapsto j) - f_i(x))}{\sum_{h=1,h\neq i}^{g} exp(f_i(i \mapsto h) - f_i(x))} \tag{3}$$

5.3.3 Gratification Function

Sometimes the order in which changes occur makes a difference for the desirability of the network. Such differences cannot be represented by the objective function. So, we need another function, called gratification function.

The *gratification function* $g_i(x, j)$ for developer i is the value attached to this developer (in addition to what follows from the objective function) to the act of changing the tie variable x_{ij} from i to j, given the current network configuration x.

In the case that a gratification function is included, the developer i chooses to change x_{ij} for that other developer j for whom the following value [50] is the highest:

$$f_i(x(i \mapsto j)) + g_i(x, j) + U(j) \tag{4}$$

And we will have

$$p_{ij}(x) = \frac{exp(f_i(i \mapsto j) + g_i(x, j) - f_i(x))}{\sum_{h=1, h \neq i}^{g} exp(f_i(i \mapsto h) + g_i(x, j) - f_i(x))} \tag{5}$$

5.3.4 Markov Chain Transition Rate Matrix

The components of the developers-oriented model, described above, define a continuous-time Markov chain on the space χ of all directed graphs on this set of g developers. This Markov chain is used to estimate the model parameters stochastically, instead of calculating them exactly, which is not possible for us.

This Markov chain has a transition rate matrix. The transition rate matrix (also called intensity matrix), for this model is given by

$$\begin{aligned} q_{ij}(x) &= \lim_{dt \to 0} \frac{1}{dt} \mathrm{P}\big(X(t + dt) = X(i \mapsto j) | X(t) = x\big) \\ &= \lambda_i(x) p_{ij}(x) \end{aligned} \tag{6}$$

Expression (6) shows the rate at which developer i makes ministeps, multiplied by the probability that he changes the arc variable X_{ij}, if he makes a ministep.

5.3.5 Markov Chain Simulation

Our Markov chain can be simulated by repeating the following steps [50]:

Start at time t with directed graph x.

1. Define $\lambda_+ = \sum_{i=1}^{g} \lambda_i(x)$ and let Δt be a random variable with the exponential distribution with parameter $\lambda_+(x)$.
2. The developer i is chosen randomly with probabilities $\frac{\lambda_i(x)}{\lambda_+(x)}$
3. Given this i, choose developer j randomly with probabilities

$$p_{ij}(x) = \frac{exp(f_i(i \mapsto j) + g_i(x, j) - f_i(x))}{\sum_{h=1, h \neq i}^{g} exp(f_i(i \mapsto h) + g_i(x, j) - f_i(x))}$$

4. Now change t to $t + \Delta t$ and change x_{ij} to $(1 - x_{ij})$

5.4 Model Specification

In the previous sections, we described why we need a statistical model to describe the longitudinal trends, rather than numerical measurements. We then described a framework for such a model; namely the developer-oriented model. We described the components of this statistical model; the rate function; the objective function, and the gratification function.

In this section, we describe what network effects can be measured and tested in these functions. Specifically, we need to supply these functions with a specific model for the rate, objective and gratification functions. These functions depend on unknown parameters that need to be estimated based on the data. The estimation is explained in section 5.5.

In the following, we explain what model effects can be included in the objective and gratification functions, to be estimated later.

5.4.1 Objective Function

The following weighted sum represents the objective function:

$$f_i(\beta, x) = \sum_{k=1}^{L} \beta_k s_{ik}(x) \tag{7}$$

Parameters $\beta = (\beta_1, ..., \beta_L)$ is to be estimated. Functions $s_{ik}(x)$ can be the following [50]:

5.4.1.1 Structural Effects For the structural effects, the following can be used. The mathematical formulas for the following effects are included in the appendix section 3.

1. The reciprocity effect, which reflects the tendency toward reciprocation of connections. A high value for its model parameter will indicate a high tendency of developers for reciprocated interactions.

2. The closure effects (e.g. in friendship networks, it means, friends of friends tend to become friends) For example,

(a) Transitive triplets effect, which models the tendency toward network closure. It reflects the preference of developers to be connected to developers with similar outgoing ties.

(b) Transitive ties, which reflects network embeddedness. It is similar to transitive triplets effect, however, it only count for the existence of at least one two-path $i \to h \to j$, rather than counting how many such two-paths exists.

(c) Balance effect, which reflects how similar the outgoing ties of developer i are to the outgoing ties of the other developers to whom i is connected.

(d) Number of developers at distance two, which reflects and is an inverse measure of, network closure.

3. Three-cycles, may be interpreted as the tendency toward local hierarchy. It is similar to reciprocity defined for three developers, and is the opposite of hierarchy.

4. Betweenness, which reflects brokerage, is the tendency of developer i to position himself between other not-directly-connected developers. represents brokerage

5. Density effect which reflects tendency to have connections at all (i.e. the out-degree of a developer i).

6. Activity, which reflects the tendency of developers with high in-degree/out-degrees to send out more outgoing connections because of their current high in-degree/out-degree.

7. Covariate effects: Developers' covariates may influence the formation or termination of ties. For example:

   (a) Covariate V-related popularity, which reflects the sum of covariate V for all developers to whom developer i is connected

   (b) Covariate V-related activity, which reflects the developer i's out-degree multiplied by his covariate V value.

   (c) Covariate V-related dissimilarity, which reflects the sum of differences in covariate V values' between developer i and all developers to whom developer i is connected.

We use the following developer attributes as covariates:

- (Covariate V1) Developer's level of contribution/activity (e.g. code commits per month, or mailing list posts per month)
- (Covariate V2) Developer's level of privilege/prestige (e.g. admin privilege-holder vs. core contributor vs. marginal developer/user)
- (Covariate V3) Developer's level of negative experience (e.g. number of rejected pull requests: especially in forks for more community-driven development)
- (Covariate V4) Developer's age (either birth age, (young 20-year-old developer vs. older 50-year-old developer) or their seniority as a development community member (i.e. how long they have been in the community)
- (Covariate V5) Developer's propensity for short-responses vs. long-responses. (under-communicator vs. over-communicator)
- (Covariate V6) Developer's communication sentiment (generally bitter in communication known as "a jerk", or has a positive communication prose)
- (Covariate V7) Developer's engagement style (i.e. dive-bomber-style contributor who jumps in, overcommits and leave, vs. a steady and increasingly engaged contributor who starts off slow and grows his/her commitment gradually)

8. in-in degree assortativity, which reflects the tendency of developers with high in-degree to be connected to other developers with high in-degrees

9. in-out degree assortativity, which reflects which reflects the tendency of developers with high in-degree to be connected to other developers with high out-degrees

10. out-in degree assortativity, which reflects which reflects the tendency of developers with high out-degree to be connected to other developers with high in-degrees

11. out-out degree assortativity, which reflects which reflects the tendency of developers with high out-degree to be connected to other developers with high out-degrees

A summary of expectation for model effect, based on intuition, is summarized in Table 8.

5.4.1.2 Behavior-related Effects

Properties of developers can be called their behavior. For a behavior-related variable $s^z_{ik}(x,z)$, we can estimate the following effects.

Let's denote a dependent behavior variable as Z.

1. Shape, which reflects the drive toward high values on the variable Z.
2. Similarity:
   (a) Average similarity, which reflects the preference of the developer i to be similar to his alters, with regard to the behavior variable Z
   (b) Total similarity, which reflects the preference of the developer i to be similar to his alters. The total influence of the alters depends on the number of the alters
3. Average alter, which reflects that the developers who have alters with higher average values of Z, have a stronger tendency to having high values of Z.

5.4.2 Gratification Function

The following weighted sum represents the gratification function:

$$g_i(\gamma, x, j) = \sum_{h=1}^{H} \gamma_h r_{ijh}(x) \tag{8}$$

Some possibile functions $r_{ijh}(x)$ are the following [50]:

1. Breaking off a reciprocated tie:
   $r_{ij1}(x) = x_{ij}x_{ji}$
2. Number of indirected links for creating a new tie:
   $r_{ij2}(x) = (1 - x_{ij}) \sum_h x_{ih}x_{hj}$
3. Effect of dyadic covariate W on breaking off a tie:
   $r_{ij3}(x) = x_{ij}w_{ij}$

Table 8: Expectations for significance of the statistical model covariates based on intuition. Note that * indicates significantly-related; +/- indicates positively/negatively related.

Model parameter	**U.F. Pers. Dif.**	**U.F. More Comm.**	**H.F. Tech. Dif.**
Reciprocity	*		*
Transitive triplets	*		*
Number of developers at distance two		*-	
Three-cycles	*		*
out-out degree assortativity		*+	
Covariate-V1 [Developer's contribution level]-related popularity	*		*
Covariate-V2 [Developer's privilege level]-related dissimilarity		*-	*
Covariate-V3 [Developer's negative experience level]-related popularity		*	
Covariate-V3 [Developer's negative experience level]-related dissimilarity	*		
Covariate-V4 [Developer's age-seniority level]-related popularity	*		
Covariate-V4 [Developer's age-seniority level]-related dissimilarity	*	*	
Covariate-V5 and V6[Developer's under/over communicator & sentiment history]-related activity and dissimilarity	*		
Covariate-V7 [Developer's graduality level]-related activity	*	*-	
Covariate-V7 [Developer's graduality level]-related dissimilarity	*	*-	

5.5 Markov Chain Monte Carlo (MCMC) Estimation

The described statistical model for longitudinal analysis of open source software development communities is a complex model and cannot be exactly calculated, but it can be stochastically estimated. We can simulate the longitudinal evolution, and estimate the model based on the simulations. Then we can choose an estimated model that has a good fit to the network data. The following section described the simulation and estimation procedures.

Network evolution can be simulated using a MCMC estimation method. The method of moments (MoM) can be used in the following way [50]:

Let $x^{obs}(t_m), m = 1, ..., M$ be the observed networks, and the objective function be $f_i(\beta, x) = \sum_{k=1}^{L} \beta_k s_{ik}(x)$. In MoM, the goal is to determine the parameters β_k such that, summed over i and m, the expected values of the $s_{ik}(X(t_{m+1}))$ are equal to the observed values. In other words, MoM fits the *observed* to the *expected*. The observed target values are:

$$s_k^{obs} = \sum_{m=1}^{M-1} \sum_{i=1}^{g} s_{ik}(x^{obs}(t_{m+1})) \quad (k = 1, ..., L) \tag{9}$$

The simulations run as follows [50]:

1. The distance between two directed graphs x and y is defined as

   $$\|x - y\| = \sum_{i,j} |x_{ij} - y_{ij}| \tag{10}$$

   and let the observed distances for m = 1, ..., M-1 be

   $$c_m = \|x^{obs}(t_{m+1}) - x^{obs}(t_m)\| \tag{11}$$

2. Use the β parameter vector and the rate of change $\lambda_i(x) = 1$

3. Do the following for m = 1, ..., M-1

   (a) Define the time as 0, and start with $X_m(0) = x^{obs}(t_m)$

   (b) Simulate the model, as described in 5.3.5 until the first time point, where

   $$\|X_m(R_m) - x^{obs}(t_m)\| = c_m \tag{12}$$

4. For k = 1, ..., L, calculate the following statistics:

$$S_k = \sum_{m=1}^{M-1} \sum_{i=1}^{g} s_{ik}(X_m(R_m)) \tag{13}$$

The simulation output will be the random variables $(S, R) = (S_1, ..., S_L, R_1, ..., R_{M-1})$. The desirable outcome for the estimation is the vector parameter $\hat{\hat{\beta}}$ for which the expected and the observed vectors are the same.

5.6 Hypothesis Testing

So far, we have described what we can model, how to fit a model that explains the longitudinal changes in an open source community (or any similar network). Now, we test several hypothesis about these models. For example, we test whether a particular effect's (for example, reciprocity's) statistical significance on the model, and make meaningful statements about whether the network dynamics depends on this parameter with a p-value, indicating the significance level. Such test is explained in section 5.6.1.

Alternatively, we compare two models for two open source project communities, and make conclusions about the difference in several model effects, for example, reciprocity and three-cycles, using the method described in section 5.6.2.

Lastly, we compare several open source communities (or such networks) and test the statistical significance of the differences between their models, using the methods described in section 5.6.3. This allows us to compare within-group and across-group comparisons of longitudinal models of open-source projects. In this study, we do all three discussed forms of tests of significance. In the following subsections, the details of these tests are described.

5.6.1 Single Parameter Test

Using the actor-oriented model, t-type test of single parameters can be done using the parameter estimates and their standard errors (S.E.). For example, for testing the null hypothesis that a component k of the parameter vector β is 0, the test has approximately a standard normal distribution. For this null hypothesis:

$$H_0 : \beta_k = 0; \tag{14}$$

the t-test can be done using the following t-statistic:

$$\frac{\hat{\beta}_k}{S.E.(\hat{\beta}_k)} \tag{15}$$

Given a p-value ≤ 0.05 would indicate a strong evidence that the network dynamics depends on the corresponding parameter's effect.

5.6.2 Multi-Parameter Differences Between Groups

Given the parameter estimates $\hat{\beta}_a$ and $\hat{\beta}_b$, and their standard errors $S.E._a$ and $S.E._b$, the differences between two groups can be tested using the following t-statistics. This t-statistics, under the null hypothesis of equal parameters, has an approximately standard normal distribution:

$$\frac{\hat{\beta}_a - \hat{\beta}_b}{\sqrt{S.E._a^2 + S.E._b^2}} \tag{16}$$

5.6.3 Multivariate Analysis of Variance Between Multiple Groups

To test the statistical significance of the mean differences between groups (in our case, the different categories of forking), we may use Multivariate Analysis of Variance (MANOVA) method. In contrast to ANOVA, MANOVA method tests for the difference in two or more vectors of means.

5.7 Research Questions and Hypotheses

In section 4, we described the research objectives in general terms. Now that we have described how to model the longitudinal changes in an open source community's collaboration networks, we can be more specific about the research questions, and the hypotheses for our statistical tests. In the following, two behavior models from social sciences are briefly explained. We used these models as guiding theories for our hypotheses.

5.7.1 Behavior Models

Software development is a collective effort, so, theories about human behavior from sociology and psychology could help explain how software developers interact and collectively develop software. In particular, the following theories about human behavior are the bases for our hypotheses.

5.7.1.1 **Balance theory** [24] talks about a motivation of individuals to move toward psychological balance. Balance theory considers cognitive consistency as a motive that drives sentiment or liking relationships, as well as liking of things created by or associated with the alter in the relationship.

We may explain U.F. forked projects because of "personal differences" as a results of cognitive inconsistency between the liking relationship of other developer(s) and the software and community created by or associated with them.

5.7.1.2 **Assortativity theories**, which tries to explain collaboration, and people's preference for interacting with others who are similar to them in some way. This is related to reciprocity, in forms of contingent, direct, and indirect reciprocity. Homophily and Heterophily, which affect diversity of networks, are also closely related to assortative selection.

We may explain U.F. forked projects because of "more community-driven development" as a by-result of developers' homophile collaborator selection.

5.7.2 Research Questions and Hypotheses

The following are the null hypotheses for across-group comparisons using the Multivariate Analysis of Variance Between Multiple Groups tests. Our prediction is that patterns exist in the data, and those patterns will show statistically significant differences between different categories of forks. Statistically speaking, we'll be able to reject the following null hypotheses:

5.7.3 Null Hypothesis # 1:

There are no statistically significant differences in vectors of means of the developer-oriented model of longitudinal changes of software projects in U.F. category and No.F. category.

5.7.4 Null Hypothesis # 2:

There are no statistically significant differences in vectors of means of the developer-oriented model of longitudinal changes of software projects in H.F. category and No.F. category.

5.7.5 Null Hypothesis # 3:

There are no statistically significant differences in vectors of means of the developer-oriented model of longitudinal changes of software projects in U.F. category and H.F. category.

If the data does not exhibit statistically significant differences, then our study fails, and our assumptions cannot be proven. We argue that based on the guiding theories and the collaboration data we gathered, if the forking reasons were socially-related, they should be reflected in the data, and so, should be reflected in the longitudinal model of their collaboration network evolution. This is a conjecture, and our study will try to show that this conjecture is true, using sound statistical modeling.

5.8 Sentiment Analysis

5.8.1 Data Collection

A statistical time series analysis of the sentiments for the contents of the messages sent and received by the developers in the 10-month run-up to the fork was completed. The messages were collected from the projects' mailing list messages, and contained all messages by all developers for that time period.

5.8.2 Methodology

We used the R (Statistical Computing) [44] package SentimentAnalysis [18][43] to analyze the sentiments of the cleaned data, and each message was assigned a sentiment score between [-1, +1], with negative values indicating negative sentiments, 0 for neutral, and positive value representing positive sentiments.

5.8.3 Results

The time series shown in Figures 3 and 4 show the daily average sentiments of the developers' messages.

Figures 5-7 show the decomposed time series into seasonal, trend and irregular components using moving averages, for all projects.

To detect outliers, we used the R (Statistical Computing) [44] package tsoutliers [33] for outlier detection by estimating outlier effects for four types

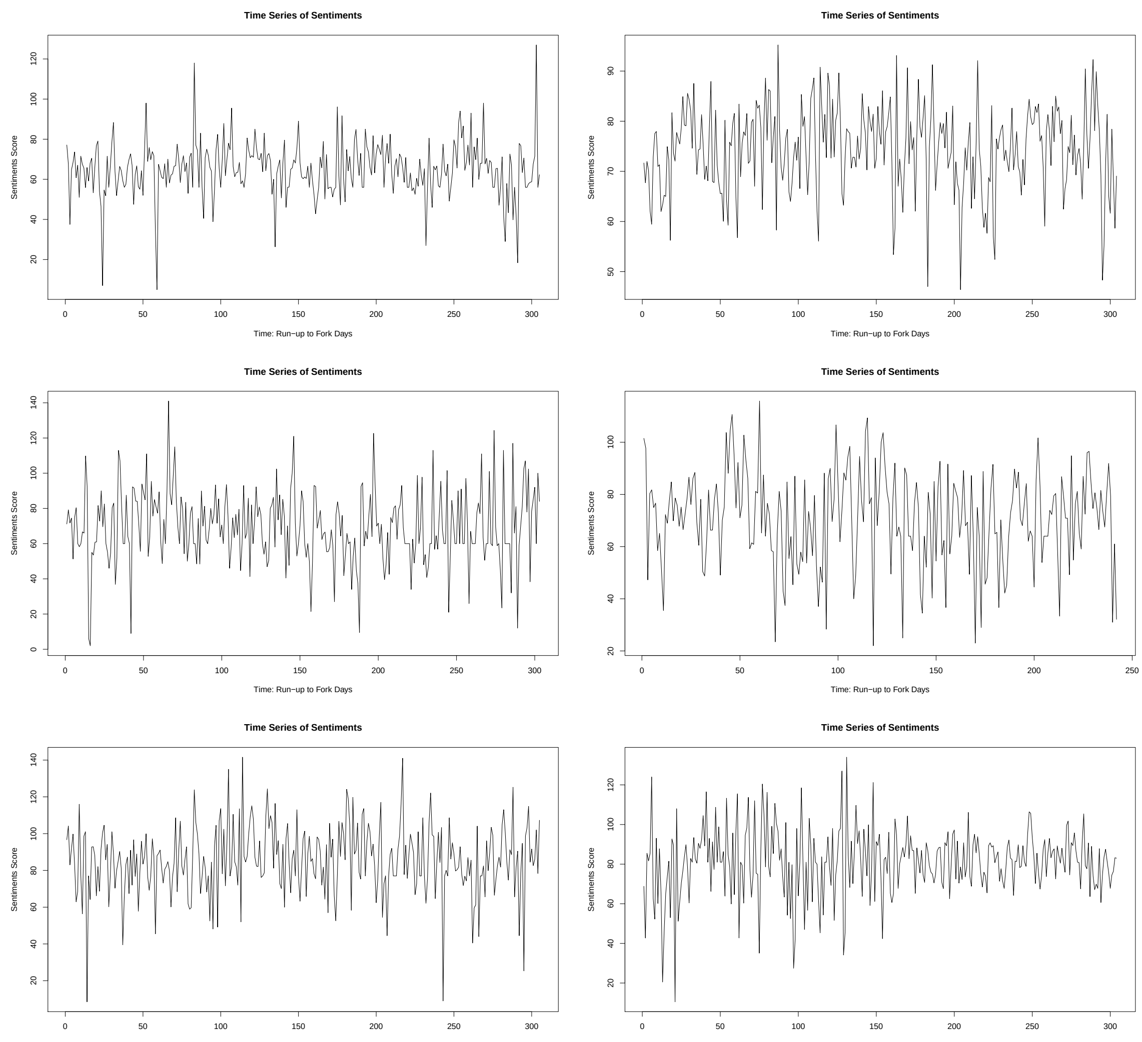


Figure 3: Time Series of sentiments in developers' messages for projects (From top to bottom, left to right) Kamailio, ffmpeg, Amarok, Apache CouchDB, Pidgin, MPlayer.

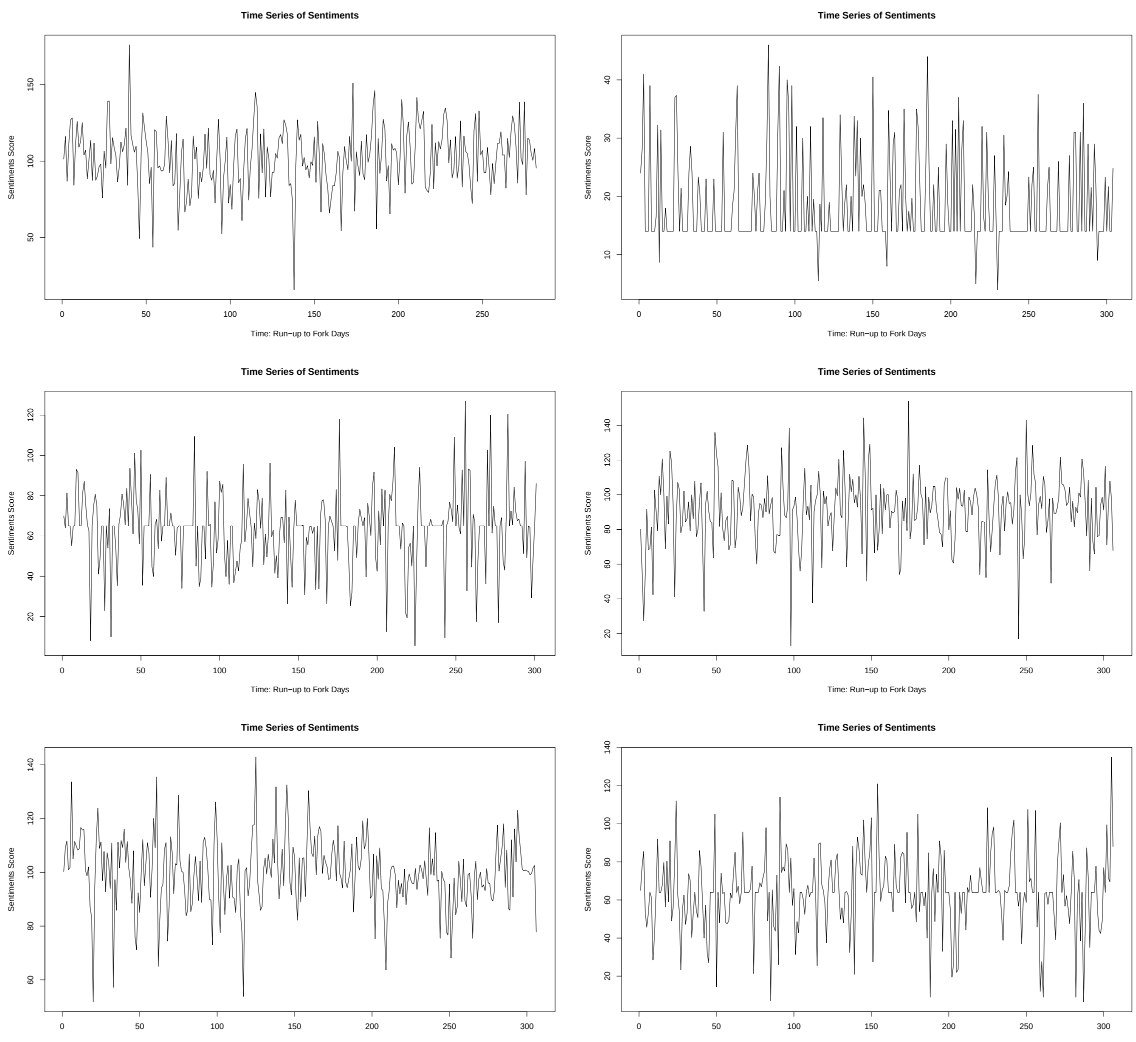


Figure 4: Time Series of sentiments in developers' messages for projects (From top to bottom, left to right) Asterisk, rdesktop, freeglut, Ceph, OpenStack Neutron, and GlusterFS

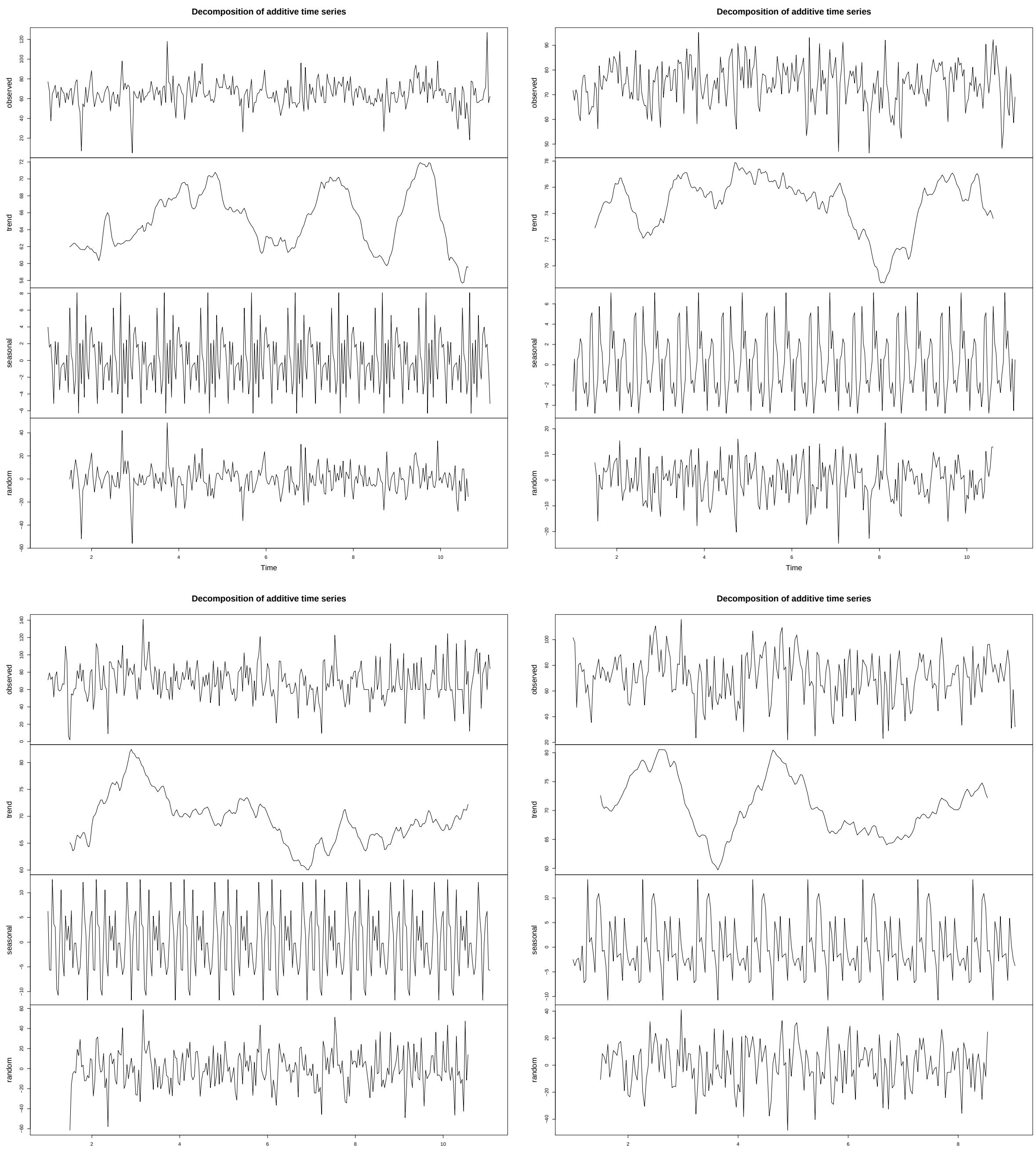


Figure 5: Time series decomposed into trend, seasonality and noise time Series for projects (From top to bottom, left to right) Kamailio, ffmpeg, Amarok, Apache CouchDB.

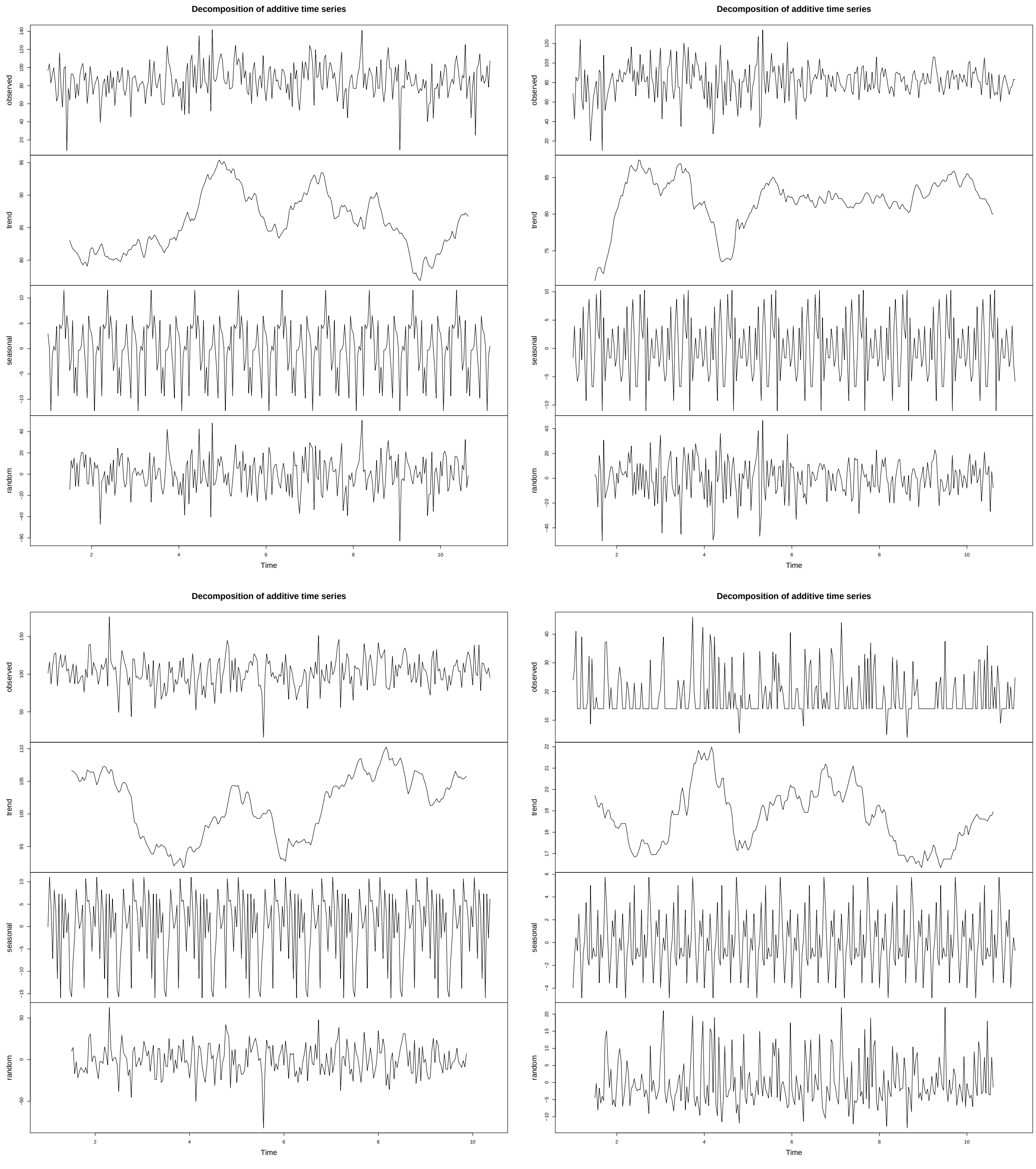


Figure 6: Time series decomposed into trend, seasonality and noise time Series for projects (From top to bottom, left to right) Pidgin, MPlayer, Asterisk, rdesktop.

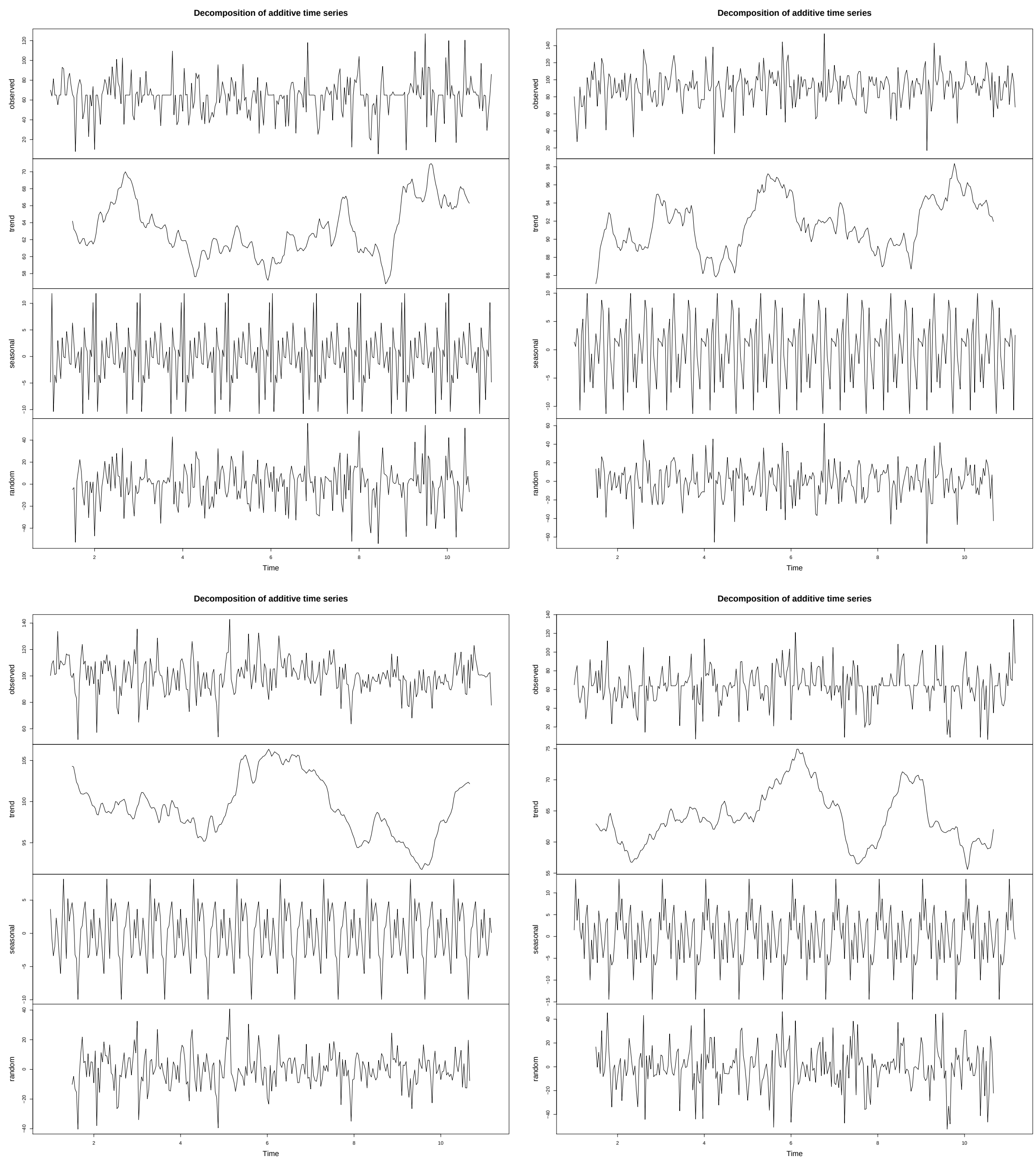


Figure 7: Time series decomposed into trend, seasonality and noise time Series for projects (From top to bottom, left to right) freeglut, Ceph, OpenStack Neutron, and GlusterFS

of outliers. We found the outliers and fitted models to the series with the outlier effects removed. Failing to adjust for outliers can result in wrong models or biased parameter estimates, and increased forecasting error. The four models for outlier effect that we looked for were Additive outlier (AO), Level shift (LS), Temporary change (TC), and Innovational outlier (IO).

Figures 8-10 show the detected outliers in the sentiments time series, and includes both the original and adjusted time series, as well as the outlier effects.

5.8.3.1 Outlier Models All the following description from [13] describes outlier models. For an ARIMA(p, d, q) process, we have

$$X_t = \frac{\theta(B)}{\alpha(B)\phi(B)} Z_t \tag{17}$$

Roots of $\theta(B), \phi(B)$ outside unit circle
$\alpha(B) = (1 - B)^d$
$Z_t \sim_{iid}$ Normal$(0, \sigma^2)$

The Observed series is represented as:

$$X_t^* = X_t + \text{ outlier effect} \tag{18}$$

$$\text{AO:} \qquad X_t^* = X_t + \omega I_t(t_1) \tag{19}$$

$$\text{LS:} \qquad X_t^* = X_t + \frac{1}{1 - B}\omega I_t(t_1) \tag{20}$$

$$\text{TC:} \qquad X_t^* = X_t + \frac{1}{(1 - \delta B)}\omega I_t(t_1) \tag{21}$$

$$\text{IO:} \qquad X_t^* = X_t + \frac{\theta(B)}{\alpha(B)\phi(B)}\omega I_t(t_1) = \frac{\theta(B)}{\alpha(B)\phi(B)}\left[Z_t + \omega I_t(t_1)\right] \tag{22}$$

5.8.3.2 Outlier Estimation The following Iterative procedure for detecting outliers, adjusting series, and fitting (seasonal) ARIMA model from [13] describes how the outliers are detected by estimation.

Obtain residuals $\hat{e}_t$ from the observed series X_t^* by applying

$$\pi(B) = \frac{\alpha(B)\phi(B)}{\theta(B)} = 1 - \pi_1 B - \pi_2 B^2 - \pi_3 B^3 - \ldots$$

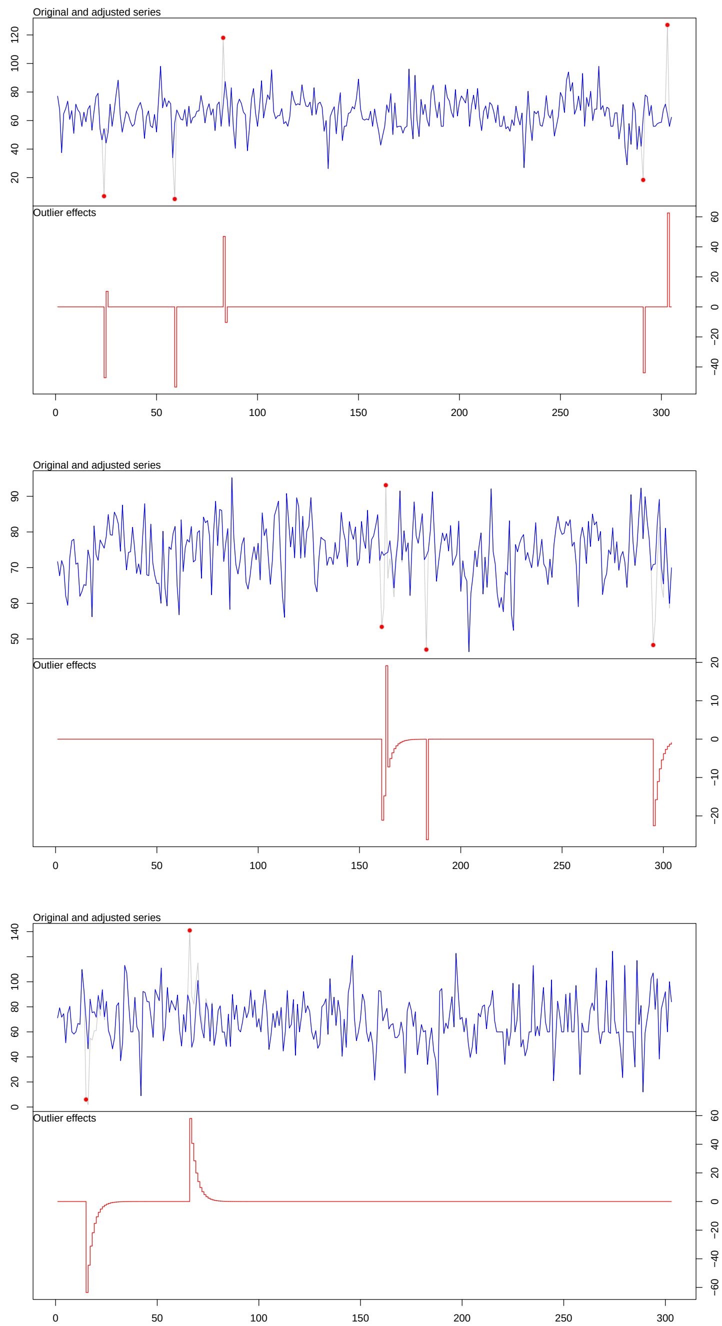


Figure 8: Time series with outliers detected for projects (From top) Kamailio, ffmpeg, Amarok. Our analysis did not find any outliers for Apache CouchDB.

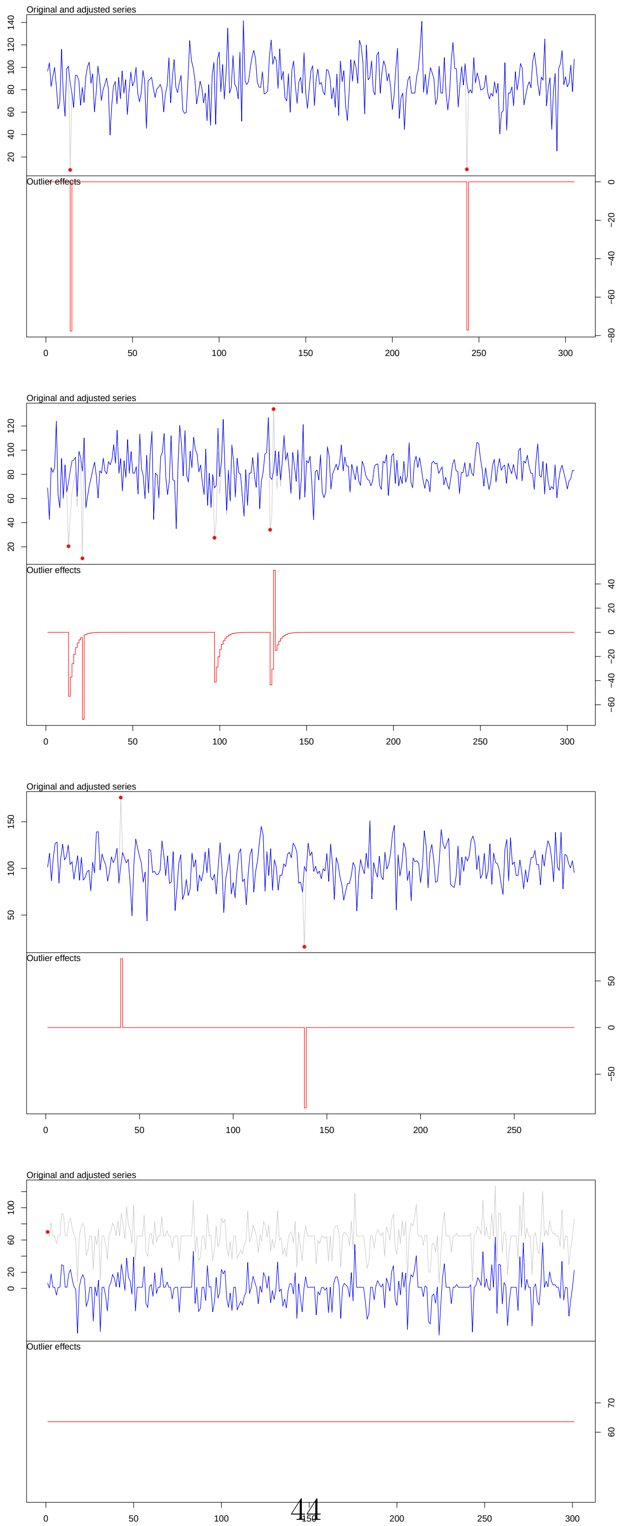


Figure 9: Time series with outliers detected for projects (From top) Pidgin, MPlayer, Asterisk, and freeglut. Our analysis did not succussfully analyse the rdesktop data.

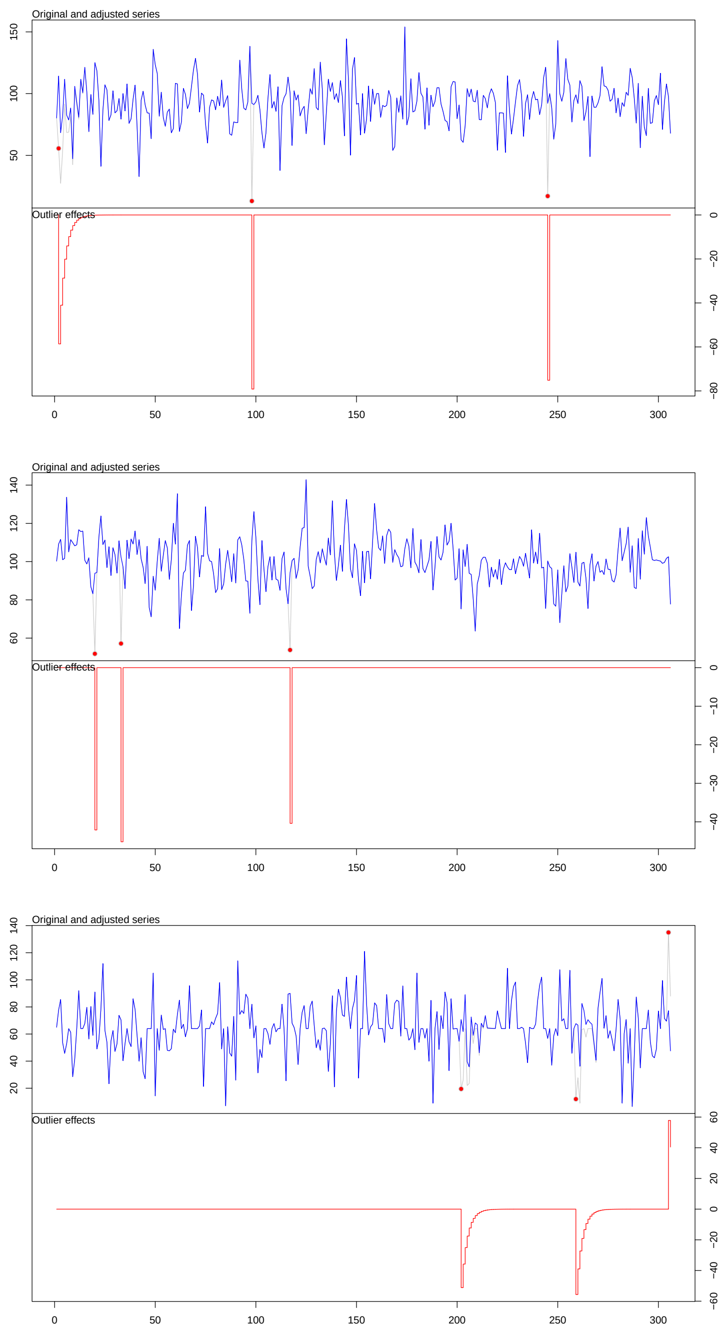


Figure 10: Time series with outliers detected for projects (From top) Ceph, OpenStack Neutron, and GlusterFS.

(Remember $X_t = \frac{\theta(B)}{\alpha(B)\phi(B)} Z_t$)

If there were no outliers, result is white noise: $\pi(B)X_t = Z_t$

When outlier present at $t = t_1$, residuals $\hat{e}_t = \pi(B)X_t^*$ for $t = t_1, \ldots, n$ reveal outlier effect.

Least-squares estimate:

$$\hat{\omega} = \frac{\sum_{t=t_1}^{n} \hat{e}_t x_t}{\sum_{t=t_1}^{n} x_t^2}$$

Divide by standard error:

$$\hat{\tau} = \frac{\hat{\omega}}{\hat{\sigma}/\sqrt{\sum_{t=t_1}^{n} x_t^2}}$$

Approximately $\sim$ Normal(0, 1)

5.8.3.3 Outlier Detection At each $t = 1, \ldots, n$, for each outlier type (AO, LS, TC, IO), Estimate outlier effect $\hat{\omega}$ and calculate $\hat{\tau}$. Large $|\hat{\tau}|$ ($> C$) indicates an outlier. Once outlier is detected, estimated effect can be subtracted to obtain adjusted series.[13]

		type	ind	time	coefhat	tstat
Project 1	1	IO	24	2007-10-28	-47.22	-4.60
	2	AO	59	2007-12-02	-53.39	-4.81
	3	IO	83	2007-12-26	47.00	4.60
	4	AO	291	2008-07-21	-43.95	-4.00
	5	AO	303	2008-08-02	62.56	5.71
		type	ind	time	coefhat	tstat
Project 2	1	TC	161	2010-10-20	-21.13	-3.23
	2	AO	163	2010-10-22	29.48	3.97
	3	AO	183	2010-11-11	-26.22	-3.64
	4	TC	295	2011-03-03	-22.58	-3.56
		type	ind	time	coefhat	tstat
Project 3	1	TC	15	2009-05-07	-63.55	-4.57
	2	TC	66	2009-06-27	58.03	4.17

Project 4: No outliers detected.

		type	ind	time	coefhat	tstat
Project 5	1	AO	14	2007-07-06	-77.62	-4.41
	2	AO	243	2008-02-20	-77.12	-4.38

		type	ind	time	coefhat	tstat
Project 6	1	TC	13	2001-06-07	-52.97	-4.94
	2	AO	21	2001-06-15	-69.10	-4.64
	3	TC	97	2001-08-30	-41.23	-3.84
	4	TC	129	2001-10-01	-43.53	-3.81
	5	AO	131	2001-10-03	72.68	4.58

		type	ind	time	coefhat	tstat
Project 7	1	AO	40	2006-10-14	73.85	3.92
	2	AO	138	2007-01-20	-86.15	-4.57

		type	ind	time	coefhat	tstat
Project 9	1	LS	1	2003-06-04	63.57	59.17

		type	ind	time	coefhat	tstat
Project 10	1	TC	2	2012-04-24	-58.61	-4.54
	2	AO	98	2012-07-29	-79.15	-4.41
	3	AO	245	2012-12-23	-75.15	-4.19

		type	ind	time	coefhat	tstat
Project 11	1	AO	20	2013-05-12	-42.11	-3.87
	2	AO	33	2013-05-25	-45.18	-4.17
	3	AO	117	2013-08-17	-40.40	-3.71

		type	ind	time	coefhat	tstat
Project 12	1	TC	202	2012-11-10	-51.17	-3.86
	2	TC	259	2013-01-06	-55.65	-4.20
	3	TC	305	2013-02-21	57.81	3.83

5.8.4 Discussion

5.9 Validation

5.9.1 Qualitative Study: Interviews and Survey

To validate what our quantitative approach finds, and to account and check for possible confounding factors, we need to compare it to what people remember of the situation. This validation check requires interviewing and surveying people from the studied forked projects. Semi-structured interviews need to be conducted, with as many developers from the forked projects, till the interviewers reach a point of saturation (i.e., when no new information is gained by doing more interviews), as possible. These semi-structured interviews will be recorded, transcribed, and coded according to the statistical model's covariates, to find overlapping and common patterns.

5.9.2 Cross-Validation

To test and validate our quantitative findings, we will model projects with "unknown" (or treated as "unknown") forking history using the same longitudinal modeling method.

The new model can then be compared to the "known" models in each forking category, using the ANOVA test. This comparison can provide new insights as to which category of forking reasons is the likely reason for forking or not-forking of the "unknown" projects. In this way, we may extrapolate about new projects' collaboration patterns.

6 Results

Table 9: Estimates parameters in the model for Project Kamailio

Effect	par.	(s.e.)	t stat.
Rate 1	1.26	(0.30)	-
Rate 2	1.99	(0.40)	-
Rate 3	2.08	(0.42)	-
Rate 4	2.51	(0.48)	-
Rate 5	2.93	(0.56)	-
Rate 6	2.69	(0.52)	-
Rate 7	2.18	(0.46)	-
Rate 8	2.45	(0.51)	-
Rate 9	2.77	(0.50)	-
outdegree (density)	–2.11***	(0.25)	–8.38
reciprocity	1.19†	(0.61)	1.95
transitive triplets	2.22**	(0.73)	3.04
3-cycles	–1.10†	(0.62)	–1.78
out-out degree$\hat{(}1/2)$ assortativity	–1.00*	(0.50)	–2.01
devSeniority alter	0.00	(0.00)	1.39
devSeniority ego	0.00*	(0.00)	1.99
devSeniority ego x devSeniority alter	0.00	(31.61)	0.00
devScActivity alter	–0.00	(0.01)	–0.33
devScActivity ego	0.03	(0.02)	1.50
devScActivity ego x devScActivity alter	–0.00	(0.00)	–1.54
int. devScActivity ego x devSeniority ego	–0.00	(31.61)	–0.00

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 0.38,
overall maximum convergence ratio 1.18.

Table 10: Estimates parameters in the model for Project ffmpeg

Effect	par.	(s.e.)	t stat.
Rate 1	5.21	(0.54)	-
Rate 2	4.52	(0.43)	-
Rate 3	4.52	(0.46)	-
Rate 4	5.62	(0.62)	-
Rate 5	4.08	(0.40)	-
Rate 6	4.69	(0.54)	-
Rate 7	4.39	(0.53)	-
Rate 8	27.77	(5.86)	-
Rate 9	3.46	(0.32)	-
outdegree (density)	–2.20***	(0.06)	–39.64
reciprocity	–2.52*	(1.27)	–1.98
transitive triplets	3.63***	(0.85)	4.27
3-cycles	–2.80**	(1.05)	–2.67
out-out degree$\hat{(}1/2)$ assortativity	–1.42***	(0.40)	–3.56
devSeniority ego	N.A.	(N.A.)	-
devScActivity ego	N.A.	(N.A.)	-
int. devScActivity ego x devSeniority ego	N.A.	(N.A.)	-

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 1.3,
overall maximum convergence ratio 3.45.

Table 11: Estimates parameters in the model for Project Amarok

Effect	par.	(s.e.)	t stat.
Rate 1	1.29	(0.33)	-
Rate 2	2.15	(0.63)	-
Rate 3	2.12	(0.63)	-
Rate 4	5.09	(1.79)	-
Rate 5	3.96	(1.16)	-
Rate 6	6.21	(2.27)	-
Rate 7	1.57	(0.39)	-
Rate 8	0.62	(0.20)	-
Rate 9	0.68	(0.21)	-
outdegree (density)	-4.33^{***}	(0.25)	-17.21
reciprocity	-3.47	(6.23)	-0.56
transitive triplets	2.91^{***}	(0.85)	3.42
3-cycles	-2.58	(8.58)	-0.30
out-out degree$\hat{(}1/2)$ assortativity	0.14	(0.38)	0.37
devSeniority alter	-0.00	(0.00)	-1.01
devSeniority ego	0.00	(0.00)	1.37
devSeniority ego x devSeniority alter	0.00	(31.61)	0.00
devScActivity alter	-0.01	(0.01)	-1.02
devScActivity ego	0.01^{**}	(0.01)	2.79
devScActivity ego x devScActivity alter	-0.00	(0.00)	-0.26
int. devScActivity ego x devSeniority ego	-0.00	(31.61)	-0.00

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 1.3,
overall maximum convergence ratio 2.03.

Table 12: Estimates parameters in the model for Project Apache CouchDB

Effect	par.	(s.e.)	t stat.
Rate 1	1.31	(0.33)	-
Rate 2	2.24	(0.67)	-
Rate 3	2.21	(0.68)	-
Rate 4	5.36	(1.94)	-
Rate 5	4.11	(1.19)	-
Rate 6	6.40	(2.35)	-
Rate 7	1.62	(0.41)	-
Rate 8	0.65	(0.21)	-
Rate 9	0.70	(0.22)	-
outdegree (density)	-4.37^{***}	(0.26)	-17.02
reciprocity	-3.97	(7.73)	-0.51
transitive triplets	2.98^{***}	(0.84)	3.53
3-cycles	-1.57	(5.86)	-0.27
out-out degree$\hat{(}1/2)$ assortativity	0.14	(0.40)	0.34
devSeniority alter	-0.00	(0.00)	-0.89
devSeniority ego	0.00	(0.00)	1.20
devSeniority ego x devSeniority alter	-0.00	(31.61)	-0.00
devScActivity alter	-0.01	(0.01)	-1.04
devScActivity ego	0.02^{**}	(0.01)	3.02
devScActivity ego x devScActivity alter	-0.00	(0.00)	-0.36
int. devScActivity ego x devSeniority ego	-0.00	(31.61)	-0.00

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 0.56,
overall maximum convergence ratio 1.38.

Table 13: Estimates parameters in the model for Project Pidgin

Effect	par.	(s.e.)	t stat.
Rate 1	1.31	(0.34)	-
Rate 2	2.22	(0.67)	-
Rate 3	2.17	(0.67)	-
Rate 4	5.20	(1.88)	-
Rate 5	4.07	(1.22)	-
Rate 6	6.35	(2.40)	-
Rate 7	1.59	(0.40)	-
Rate 8	0.63	(0.20)	-
Rate 9	0.70	(0.22)	-
outdegree (density)	-4.38^{***}	(0.26)	-16.91
reciprocity	-4.93	(10.13)	-0.49
transitive triplets	2.96^{***}	(0.85)	3.47
3-cycles	-2.87	(9.06)	-0.32
out-out degree$\hat{(}1/2)$ assortativity	0.17	(0.40)	0.43
devSeniority alter	-0.00	(0.00)	-0.72
devSeniority ego	$0.00^{\dagger}$	(0.00)	1.83
devSeniority ego x devSeniority alter	0.00	(31.61)	0.00
devScActivity alter	-0.02	(0.01)	-1.27
devScActivity ego	0.01^{*}	(0.01)	2.57
devScActivity ego x devScActivity alter	-0.00	(0.00)	-0.22
int. devScActivity ego x devSeniority ego	-0.00	(31.61)	-0.00

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 1.21,
overall maximum convergence ratio 1.62.

Table 14: Estimates parameters in the model for Project MPlayer

Effect	par.	(s.e.)	t stat.
Rate 1	1.28	(0.33)	-
Rate 2	2.17	(0.65)	-
Rate 3	2.11	(0.66)	-
Rate 4	5.02	(1.77)	-
Rate 5	3.99	(1.16)	-
Rate 6	6.07	(2.21)	-
Rate 7	1.57	(0.39)	-
Rate 8	0.63	(0.19)	-
Rate 9	0.69	(0.21)	-
outdegree (density)	-4.35^{***}	(0.25)	-17.18
reciprocity	-6.27	(9.93)	-0.63
transitive triplets	2.79^{***}	(0.81)	3.44
3-cycles	-2.40	(4.89)	-0.49
out-out degree$\hat{(}1/2)$ assortativity	0.24	(0.36)	0.65
devSeniority alter	-0.00	(0.00)	-0.68
devSeniority ego	0.00	(0.00)	1.58
devSeniority ego x devSeniority alter	0.00	(31.61)	0.00
devScActivity alter	-0.02	(0.01)	-1.29
devScActivity ego	0.01^{*}	(0.00)	2.47
devScActivity ego x devScActivity alter	-0.00	(0.00)	-0.30
int. devScActivity ego x devSeniority ego	-0.00	(31.61)	-0.00

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 0.8,
overall maximum convergence ratio 1.16.

Table 15: Estimates parameters in the model for Project rdesktop

Effect	par.	(s.e.)	t stat.
Rate 1	1.77	(1.02)	-
Rate 2	0.86	(0.43)	-
Rate 3	1.47	(0.70)	-
Rate 4	1.43	(0.54)	-
Rate 5	1.52	(0.83)	-
Rate 6	1.26	(0.66)	-
Rate 7	1.54	(0.85)	-
Rate 8	1.33	(0.68)	-
Rate 9	0.37	(0.39)	-
outdegree (density)	-2.78^{***}	(0.77)	–3.61
reciprocity	–1.35	(1.30)	–1.04
transitive triplets	–0.27	(1.03)	–0.26
3-cycles	–3.30	(4.64)	–0.71
out-out degree$\hat{(}1/2)$ assortativity	$1.27^{\dagger}$	(0.65)	1.95
devSeniority ego	N.A.	(N.A.)	-
devScActivity ego	N.A.	(N.A.)	-
int. devScActivity ego x devSeniority ego	N.A.	(N.A.)	-

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 0.2,
overall maximum convergence ratio 0.23.

Table 16: Estimates parameters in the model for Project freeglut

Effect	par.	(s.e.)	t stat.
Rate 1	2.32	(0.51)	-
Rate 2	3.40	(0.62)	-
Rate 3	11.19	(4.12)	-
Rate 4	2.65	(0.50)	-
Rate 5	2.36	(0.48)	-
Rate 6	2.36	(0.52)	-
Rate 7	2.07	(0.49)	-
Rate 8	1.58	(0.39)	-
Rate 9	4.06	(0.96)	-
outdegree (density)	-2.80^{***}	(0.22)	-13.01
reciprocity	0.28	(0.28)	1.00
transitive triplets	0.43^{***}	(0.09)	4.84
3-cycles	-0.18	(0.12)	-1.50
out-out degree$\hat{(}1/2)$ assortativity	0.16	(0.10)	1.61

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 0.21,
overall maximum convergence ratio 1.07.

Table 17: Estimates parameters in the model for Project Ceph

Effect	par.	(s.e.)	t stat.
Rate 1	7.03	(0.83)	-
Rate 2	5.40	(0.48)	-
Rate 3	5.64	(0.56)	-
Rate 4	6.22	(0.67)	-
Rate 5	6.77	(0.64)	-
Rate 6	22.78	(3.49)	-
Rate 7	11.85	(1.02)	-
Rate 8	7.92	(0.77)	-
Rate 9	7.19	(0.64)	-
outdegree (density)	-2.85^{***}	(0.08)	−36.95
reciprocity	1.03^{***}	(0.22)	4.76
transitive triplets	1.85^{***}	(0.25)	7.27
3-cycles	-2.77^{***}	(0.42)	−6.59
out-out degree$\hat{(}1/2)$ assortativity	-0.51^{***}	(0.11)	−4.59
devSeniority alter	0.00	(0.00)	0.48
devSeniority ego	0.00	(31.61)	0.00
devSeniority ego x devSeniority alter	−0.00	(31.61)	−0.00
devScActivity alter	-0.02^{***}	(0.00)	−3.73
devScActivity ego	0.00	(0.00)	1.41
devScActivity ego x devScActivity alter	0.00	(0.00)	0.02
int. devScActivity ego x devSeniority ego	−0.00	(31.61)	−0.00

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 0.89,
overall maximum convergence ratio 2.04.

Table 18: Estimates parameters in the model for Project OpenStack Neutron

Effect	par.	(s.e.)	t stat.
Rate 1	1.94	(0.17)	-
Rate 2	2.75	(0.24)	-
Rate 3	3.25	(0.29)	-
Rate 4	3.00	(0.25)	-
Rate 5	3.09	(0.28)	-
Rate 6	9.52	(1.15)	-
Rate 7	4.68	(0.34)	-
Rate 8	4.39	(0.33)	-
Rate 9	3.73	(0.31)	-
outdegree (density)	-3.45^{***}	(0.06)	–57.09
reciprocity	-3.80^{***}	(0.81)	–4.71
transitive triplets	2.65^{***}	(0.34)	7.80
3-cycles	-2.65^{*}	(1.10)	–2.41
out-out degree$\hat{(}1/2)$ assortativity	-0.54^{***}	(0.12)	–4.36
devSeniority alter	0.00^{*}	(0.00)	2.03
devSeniority ego	0.00^{***}	(0.00)	3.43
devSeniority ego x devSeniority alter	–0.00	(0.00)	–0.39
devScActivity alter	$0.04^{\dagger}$	(0.02)	1.94
devScActivity ego	$0.09^{\dagger}$	(0.05)	1.96
devScActivity ego x devScActivity alter	–0.02	(0.03)	–0.62
int. devScActivity ego x devSeniority ego	-0.00^{*}	(0.00)	–2.04

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 1.07,
overall maximum convergence ratio 1.83.

Table 19: Estimates parameters in the model for Project GlusterFS

Effect	par.	(s.e.)	t stat.
Rate 1	4.56	(0.74)	-
Rate 2	3.68	(0.65)	-
Rate 3	4.61	(1.01)	-
Rate 4	6.78	(1.50)	-
Rate 5	3.96	(0.64)	-
Rate 6	2.70	(0.40)	-
Rate 7	2.65	(0.51)	-
Rate 8	5.55	(1.16)	-
Rate 9	8.19	(1.77)	-
outdegree (density)	–2.78***	(0.12)	–23.25
reciprocity	–0.78	(0.68)	–1.14
transitive triplets	2.29***	(0.37)	6.15
3-cycles	–0.88†	(0.53)	–1.65
out-out degree$\hat{(}1/2)$ assortativity	–0.54**	(0.19)	–2.88
devSeniority alter	–0.00***	(0.00)	–3.38
devSeniority ego	–0.00	(0.00)	–0.13
devSeniority ego x devSeniority alter	–0.00	(31.61)	–0.00
devScActivity alter	–0.09†	(0.05)	–1.83
devScActivity ego	0.02	(0.04)	0.39
devScActivity ego x devScActivity alter	0.01	(0.01)	0.45
int. devScActivity ego x devSeniority ego	0.00	(0.00)	0.73

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 0.45,
overall maximum convergence ratio 0.71.

7 To Be Done

We proposed to model the developers interactions statistically, to find the population processes that underlie the formation, changes, and dissolution of developer communities. We expect to see distinct changes of such processes for each forking category.
Data collection for mailing list archives is completed. The issue tracking system, and source code interactions data is in the progress. Once all data is collected an cleaned, we will do the statistical modeling. Next, we will do the validation testing, which will conclude the proposed research.
We expect to find patterns specific to each category, which then may be

used to identify early warning signs of forking. The identification of such measures may inform those who are interested in the sustainability of their project community to stay informed and take action to amend undesirable dynamics.

Effect	par.	(s.e.)	t stat.
Rate 1	4.62	(0.78)	-
Rate 2	3.68	(0.65)	-
Rate 3	4.68	(1.02)	-
Rate 4	6.89	(1.54)	-
Rate 5	3.98	(0.65)	-
Rate 6	2.70	(0.39)	-
Rate 7	2.66	(0.52)	-
Rate 8	5.54	(1.15)	-
Rate 9	8.29	(1.80)	-
outdegree (density)	-2.81^{***}	(0.12)	–22.61
reciprocity	–0.63	(0.68)	–0.93
transitive triplets	2.24^{***}	(0.38)	5.85
3-cycles	–0.78	(0.52)	–1.49
out-out degree assortativity	-0.52^{**}	(0.20)	–2.66
devSeniority alter	-0.01^{***}	(0.00)	–3.58
devSeniority ego	0.00	(0.00)	0.04
devSeniority ego x devSeniority alter	–0.00	(31.61)	–0.00
devScActivity alter	$-0.08^{\dagger}$	(0.05)	–1.65
devScActivity ego	0.01	(0.04)	0.36
devScActivity ego x devScActivity alter	0.01	(0.01)	0.46
int. devScActivity ego x devSeniority ego	0.00	(0.00)	0.85

† $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$;
convergence t ratios all < 0.69,
overall maximum convergence ratio 1.03.

8 Timeline

Table 20 outlines my proposed research timeline. It also shows the mapping between research questions and the phases of the study.

Table 20 Timeline

Spring 2012	Literature review
Fall 2012	Literature review & data collection
Fall 2013	Data cleaning and wrangling
Winter 2014	Creating communication graphs
Spring 2014	Temporal visualization and temporal SNA
Fall 2014	Preliminary statistical analysis
Spring/Fall 2015	Planning and preliminary examination
Winter 2016	Prelim
Winter 2016	Data collection for issue tracking and source code
Spring/Summer 2016	Longitudinal modeling & qualitative study
Summer 2016	Thesis writing
September 2016	Defense (Hopefully)

9 Threats to Validity

The study findings may not be generalized to all FOSS projects.
First, one reason is that the projects is this research study were selected from a pool of candidate projects, based on a filtering criteria that included availability of their data. Given access, a larger number of projects as the sample size could result in a more robust investigation.
Second, we used data from online communications. The assumption that all the communication can be captured by mining repositories is intuitively imperfect, but inevitable. Third, social interactions data is noisy, and our statistical approach might be affected because of this.
Third, the statistical model we use to model the longitudinal evolution of collaboration networks is estimated stochastically, rather than being calculated exactly. The stochastic process might not always arrive at the same results. To counter this issue, we run the algorithm several times to double-check for such irregularities.

Acknowledgment

I would like to thank my academic adviser, Prof. Carlos Jensen, and my committee members for their help. I would also like to thank Derric Jacobs, of the Sociology Department at Oregon State University, for his help with search for theories and lending books to me; and also, many thanks goes to Prof. Drew Gerkey, of the Department of Anthropology at Oregon State University for his advice and pointers to theories about human behavior in social sciences. I would also like to thank the open source developers of the projects studied for making their data available, without which this study would not have been possible.

Appendices

1 Appendix A: List of all projects forked in socially-related Undesirable Forking (U.F.) Category

Table 21: List of all projects forked because of "*personal differences among the developer team*" (U.F.) [46] in chronological order. N/A means Not Applicable

Original	**Forked**	**Date**	**M.L. data accessible?**	**Collected?**
GNU Emacs	X Emacs	1991, ?	No, only after 2000	N/A
NetBSD	OpenBSD	1995, Oct	Yes, scarce, unusable	N/A
xMule	aMule	2003, Aug	No, only 2006-2007	N/A
lMule	xMule	2003, Jun	No	N/A
Sodipodi	Inkscape	2003, Nov	No	N/A
Nucleus CMS	Blog:CMS	2004, May	No, only after 09/2004	N/A
BMP	Audacious	2005, Oct	No	N/A
ntfsprogs	NTFS-3G	2006, Jul	No	N/A
OpenWRT	FreeWRT	2006, May	No, only after 10/2006	N/A
QtiPlot	SciDavis	2007, Aug	No	N/A
Kamailio	OpenSIPS	2008, Aug	Yes	Yes
Blastwave.org	OpenCSW	2008, Aug	No	N/A
jMonkeyEngine	Ardor3D	2008, Sept	Yes, scarce, unusable	N/A
Frog CMS	Wolf CMS	2009, Jul	No	N/A
Aldrin	Neil	2009, ?	No	N/A
Ffmpeg	libav	2011, Mar	Yes	Yes

Table 22: List of all projects forked because of the need for "*more community-driven development*" (U.F.) [46] in chronological order. N/A means Not Applicable

Original	**Forked**	**Date**	**M.L. data accessible?**	**Collected?**
Nethack	Slash'EM	1996, ?	No	N/A
GCC	EGCS	1997, ?	No	N/A
SourceForge	Savane	2001, Oct	No	N/A
PHPNuke	PostNuke	2001, Sum	No, not found	N/A
QTExtended	OPIE	2002, May	No	N/A
GraphicsMagick	Graphics	2002, Nov	No, only after 2003	N/A
freeglut	OpenGLUT	2004, Mar	Yes	Yes
Mambo	Joomla!	2005, Aug	No	N/A
SER	Kamailio	2005, Jun	No, only after 2006	N/A
PHPNuke	RavenNuke	2005, Nov	No, not found	N/A
Hula	Bongo	2006, Dec	No	N/A
Compiere	A Dempiere	2006, Sept	No	N/A
Compiz	Beryl	2006, Sept	No, only after 06/2007	N/A
SQL-Ledger	LedgerSMB	2006, Sept	No	N/A
Asterisk	Callweaver	2007, Jun	Yes	Yes
CodeIgniter	KohanaPHP	2007, May	No, not found	N/A
OpenOffice.org	Go-oo.org	2007, Oct	No, only after 06/2011	N/A
Mambo	MiaCMS	2008, May	No	N/A
TORCS	Speed Dreams	2008, Nov	Yes, scarce, unusable	N/A
MySQL	MariaDB	2009, Jan	Yes, too large, unusable	N/A
Nagios	Icinga	2009, May	No	N/A
Project Darkstar	RedDwarf	2010, Feb	Yes, scarce, unusable	N/A
SysCP	Froxlor	2010, Feb	No	N/A
Dokeos	Chamilo	2010, Jan	No, not found	N/A
GNU Zebra	Quagga	2010, Jul	No	N/A
rdesktop	FreeRDP	2010, Mar	Yes	Yes
OpenOffice.org	LibreOffice	2010, Sept	No, only after 06/2011	N/A
Redmine	ChiliProject	2011, Feb	Yes, scarce, unusable	N/A

2 Appendix B: Initial study: Temporal analysis using the network-specific measurement approach

In our initial study [4][5][6], which was a network-specific study, we wanted to analyze the network-specific changes that happen to the community over a given period of time, e.g., three months before and three months after the year in which the forking event happened. For this network-specific study, we measured the betweenness centrality [11] of the most significant nodes in the graph, and the graph diameter over time. Figures 15, 16, and 17 show the betweenness centralities over the 1.5 year period for the Kamailio, Amarok and Asterisk projects respectively. To do temporal analysis, we had two options; 1) look at snapshots of the network state over time, (e.g., to look at the network snapshots in every week, the same way that a video is composed of many consecutive frames), and 2) look at a period through a time window. We preferred the second approach, and looked through a time window three months wide with 1.5 month overlaps. To create the visualizations, we used a 3 months time frame that progressed six days a frame. In this way, we would have had a relatively smooth transition.

There are many ways of looking at an individual's importance/prestige/status within a network. One is called *closeness centrality*. The *farness* of a node is defined as the sum of its distances to all other nodes. The *closeness* of a node is defined as the inverse of the farness. More informally, the more central a node is the lower its total distance to all other nodes. *Closeness centrality* can be used as a measure of how fast information will spread through the network [12]. Secondly, if we are looking for people who can serve as bridges between two distinct communities, we could measure the node's *betweenness centrality*. Betweenness centralities for mediators who act as intermediate entities between other nodes are higher [12]. Third, if cross-community collaboration is the focus, we can measure *edge betweenness centrality*. Edges connecting nodes from different communities have higher edge centrality values. In the community collaboration graph, edge betweenness or stress of an edge is the number of these shortest paths that the edge belongs to, considering all shortest paths between all pairs of nodes in the graph. Fourth, one can claim that certain people in the community are more important than others, and whoever is close to them, is relatively more important than others. In graph terms,

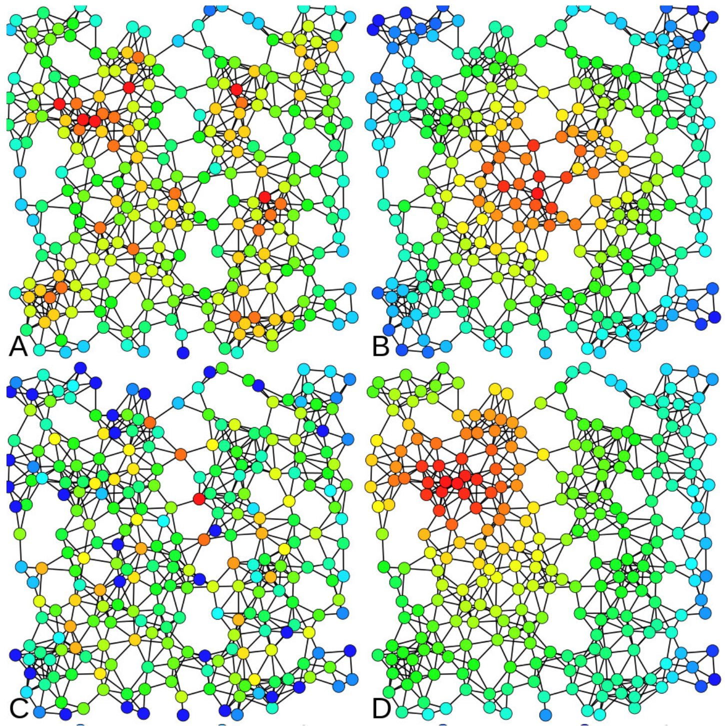


Figure 11: Heat-map color-coded examples are shown above. Nodes with higher centrality metric are colored with warmer color: red is the warmest color here. The same network is analyzed four times with the following centrality measures: A) Degree centrality, B) Closeness centrality, C) Betweenness centrality and D) Eigenvector centrality [47]

this is measured by *eigenvector centrality*, which is based on the assumption that connections to high-profile nodes contribute more to the importance of a node. Google's PageRank link-analysis algorithm [42] is a variant of the eigenvector centrality measure. In short, centrality measures have been used in several studies to identify key player in a community.
In addition to the centrality measures, we planned to look into the *resilience* of the community as well. By resilience, we mean how well the network holds its structure and form when some parts of it are deleted, added, or changed. For a graph, the resilience of a graph is a measure of its robustness to node or edge failures. This could occur for instance when an influential member of the community leaves. Many real-world graphs are resilient to random failures but vulnerable to targeted attacks. Resilience can be related to the

graph diameter: a graph whose diameter does not increase much on node or edge removal has higher resilience [12].

2.1 Visualization

Several visualization techniques and tools are used in the field of social network analysis, for instance, Gephi [8], which is a FOSS tool for exploring and manipulating networks. It is capable of handling large networks with more than 20,000 nodes and features several SNA algorithms. We used it for dynamic network visualization. We visualized the dynamic network changes using Gephi [8]. The videos[1] show how the community graph is structured, using a continuous force-directed linear-linear model, in which the nodes are positioned near or far from one another proportional to the graph distance between them. This results in a graph shape between Früchterman & Rheingold's [21] layout and Noack's LinLog [36].

2.2 Initial study results and discussion

2.2.1 Kamailio Project

Figure 14 shows four key frames from the Kamailio project's social graph around the time of their fork (the events described here are easier to fully grasp by watching the video. A node's size in a proportional to the number of interactions the node (contributor) has had within the study period and the position and edges of the nodes change if they had interactions within the time window shown, with six day steps per frame. The 1 minute and 37 seconds video shows the life of the Kamailio project between October 2007, and March 2009. Nodes are colored based on the modularity of the network. The community starts with the GeneralList as the the biggest node, and four larger core contributors and three lesser size core contributors. The big red-colored node's transitions are hard to miss, as this major contributor departs from the core to the periphery of the network (Video minute 1:02) and then leaves the community (Video minute 1:24) capturing either a conflict or retirement. This corresponds to the personal difference category of forking reasons.
Figure 15 shows the betweenness centrality of the major contributors of Kamailio project over the same time period. The horizontal axis marks the

[1]Video visualizations available at http://eecs.oregonstate.edu/~azarbaam/OSS2014/

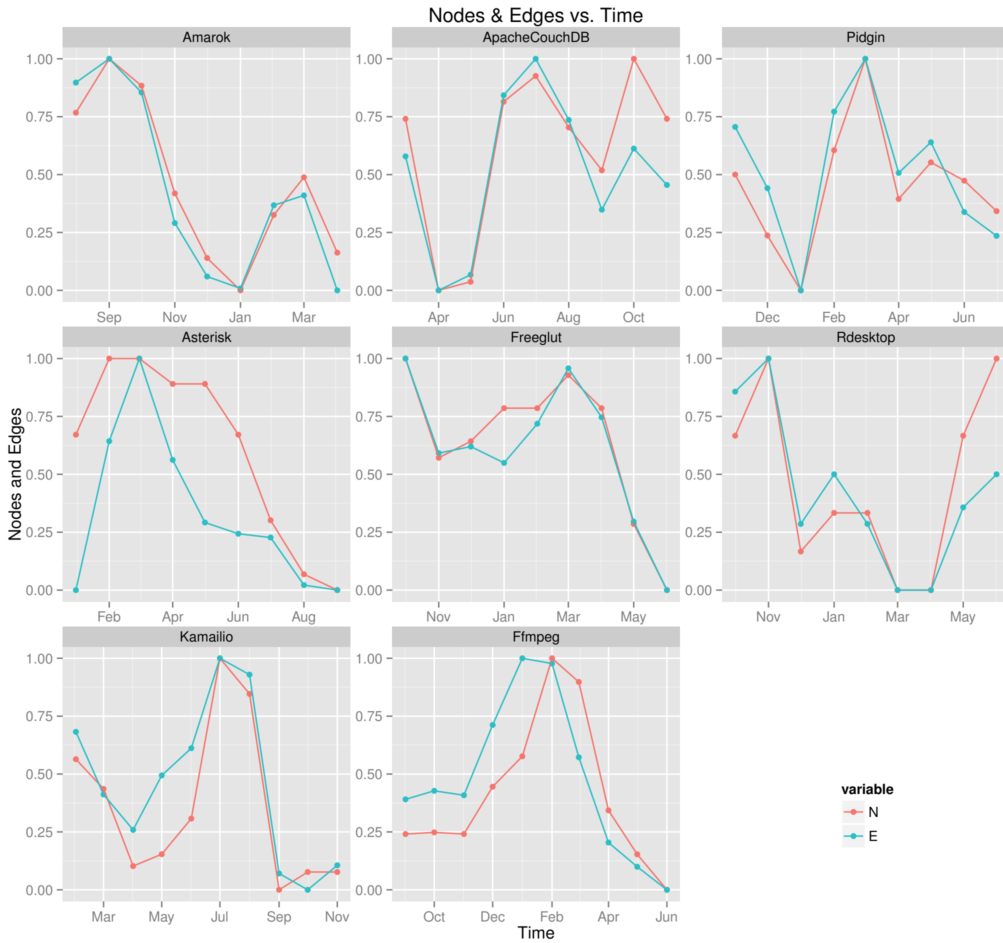


Figure 12: Nodes and Edges over Time. The number of nodes and number of edges are normalized to the range [0,1] to make comparison across projects meaningful, by emphasizing change in ratio, rather than the varying counts. Hence, the measurements were normalized for drawing this graph. The three projects in the first row belong to the *"technical differences"* forking reason category, the three projects in the second row belong to the *"more community-driven development"* forking reason category, and the two projects in the third row belong to the *"personal differences"* forking reason category.

dates, (each mark represents a 3-month time window with 1.5 months over-

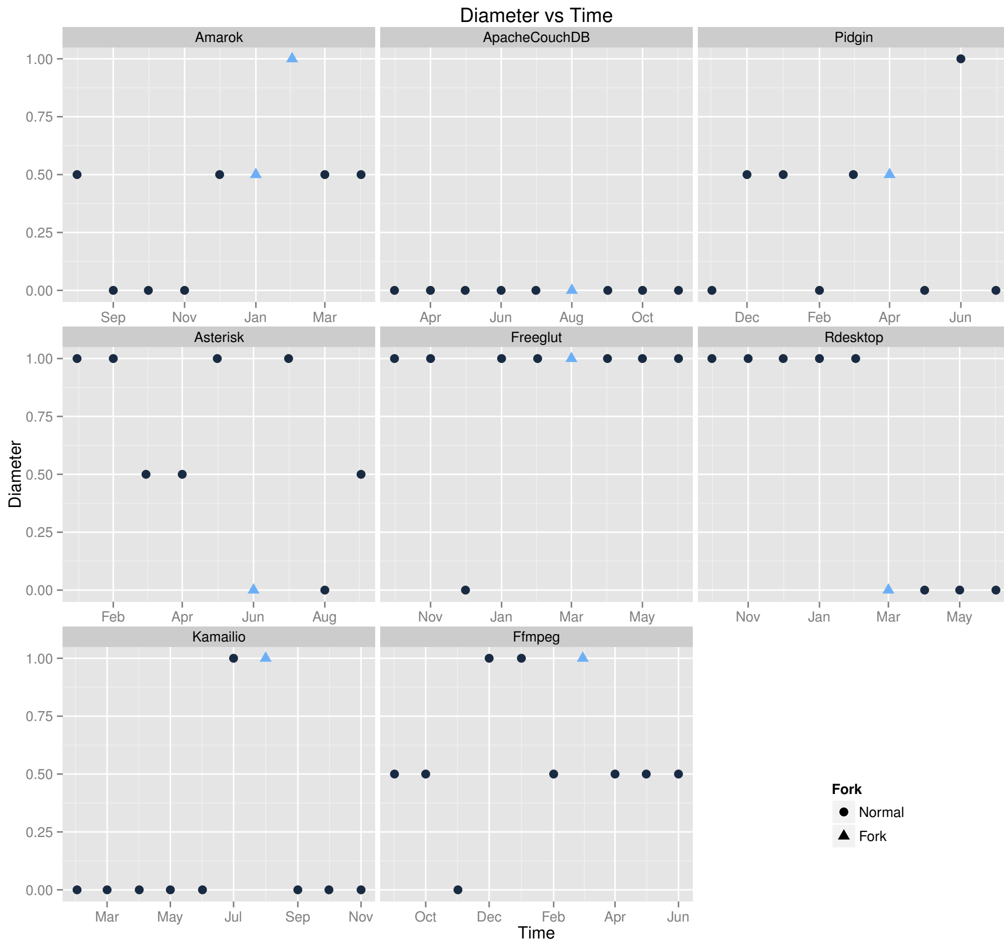
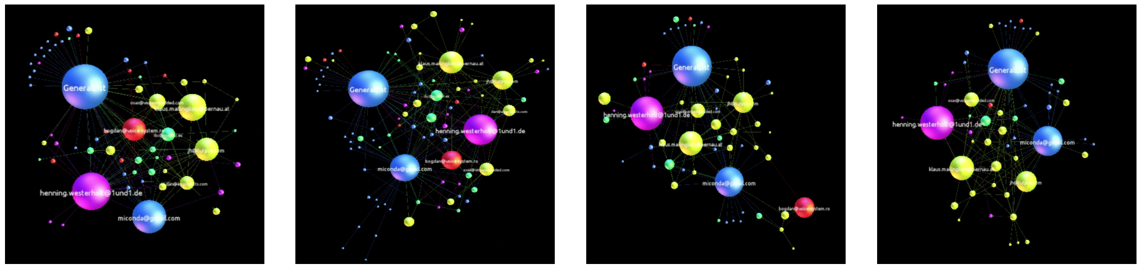


Figure 13: Diameter changes over Time. Note that the diameter measurements were normalized to the range [0,1]. The three projects in the first row belong to the *"technical differences"* forking reason category, the three projects in the second row belong to the *"more community-driven development"* forking reason category, and the two projects in the third row belong to the *"personal differences"* forking reason category.

lap). The vertical axis shows the percentage of the top betweenness centralities for each node. The saliency of the GeneralList – colored as light blue – is apparent because of its continuous and dominant presence in the stacked area chart. The chart legend lists the contributors based on the color and in



(a) (b) (c) (d)

Figure 14: Snapshots from video visualization of Kamailio's graph (Oct. 2007 - Mar. 2009) in which a core contributor (colored red) moves to the periphery and eventually departs the community.

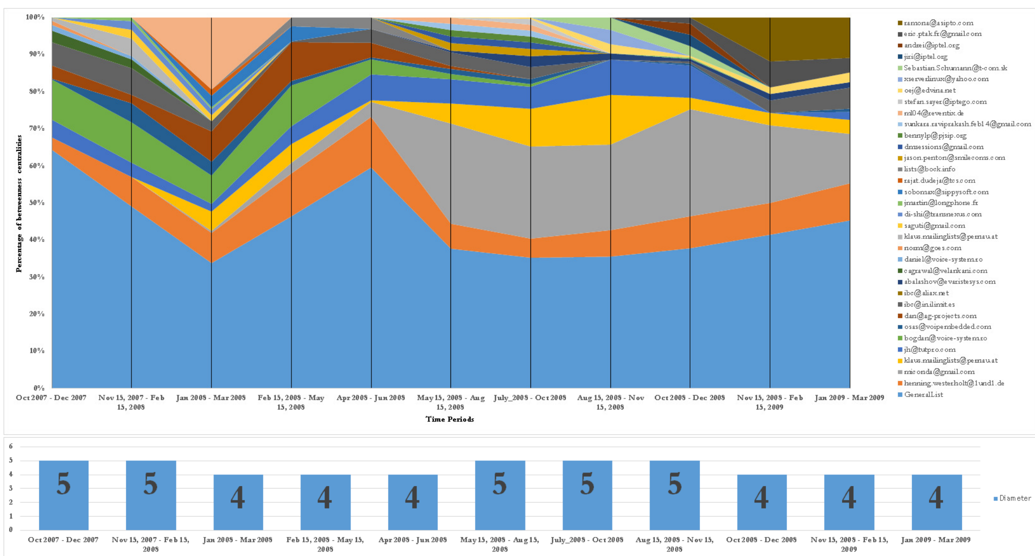


Figure 15: Kamailio top contributors' betweenness centralities and network diameter over time (Oct. 2007 to Mar. 2009) in 3-month time windows with 1.5-month overlaps

the same order of appearance on the chart starting from the bottom. Around the "Aug. 15, 2008 - Nov. 15, 2008" tick mark on the horizontal axis, several contributors' betweenness centralities shrink to almost zero and disappear. This suggests the date of fork with a month accuracy. The network diameter of the Kamailio project over the same time period is also shown in Figure

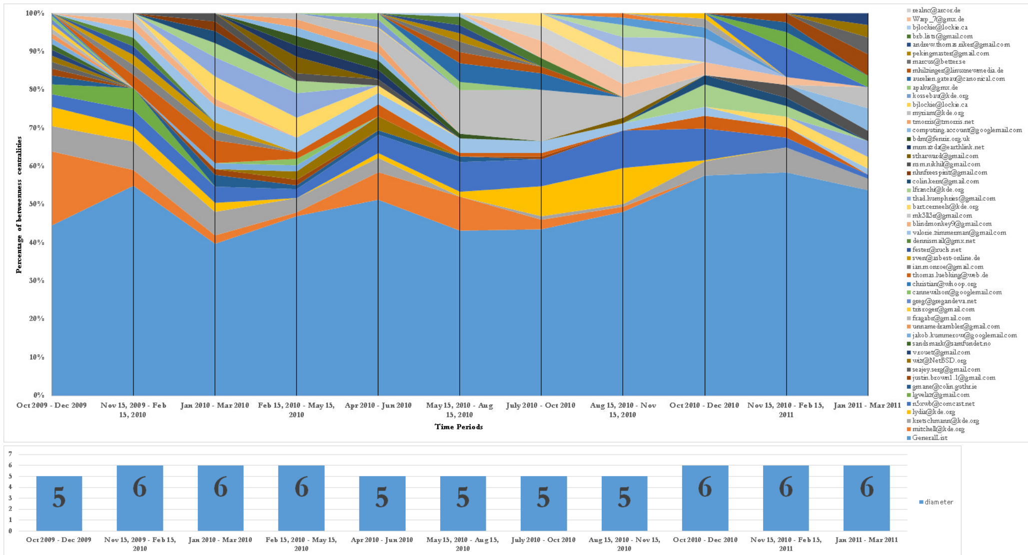


Figure 16: Amarok project's top contributors' betweenness centralities and network diameter over time between Oct. 2009 to Mar. 2011 in 3-months time windows with 1.5 months overlaps

15. An increase in the network diameter during this period is noticeable; this coincides with findings of Hannemann and Klamma [23].
This technique can be used to identify the people involved in conflict and the date the fork happened with a months accuracy, even if the rival project does not emerge immediately.

2.2.2 Amarok Project

The video for the Amarok project fork is available online[2], and the results from our quantitative analysis of the betweenness centralities and the network diameters are shown in Figure 16. The results show that the network diameter has not increased over the period of the fork, which shows a resilient network. The video shows the dynamic changes in the network structure, again typical of a non-unhealthy network, rather than of simmering conflict. These indicators suggest that the Amarok fork in 2010 belongs to the "addition of technical functionality" rationale for forking, as there are no visible social conflict.

[2]Video visualizations available at http://eecs.oregonstate.edu/~azarbaam/OSS2014/

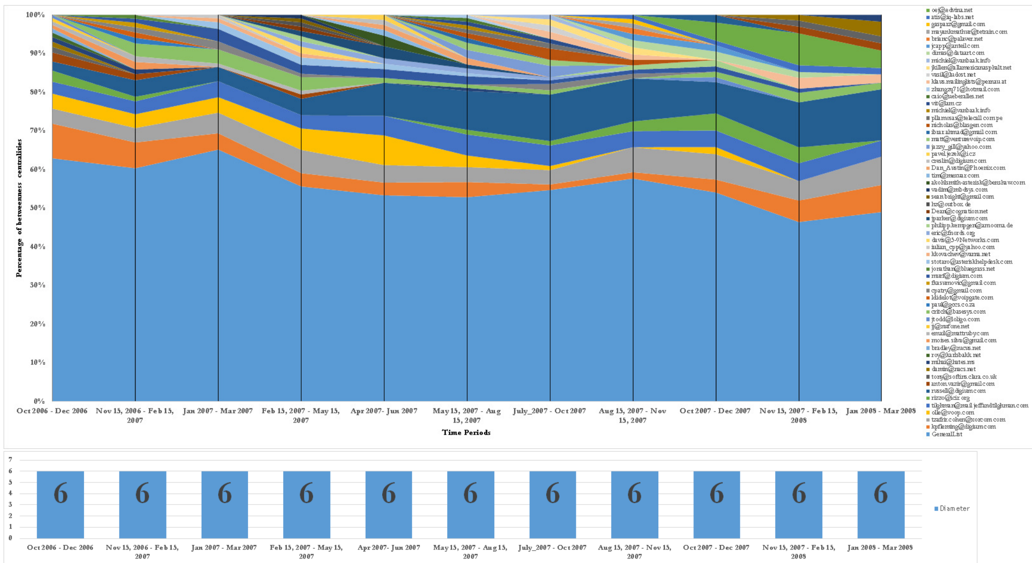


Figure 17: Asterisk project's top contributors' betweenness centralities and network diameter over time between Oct. 2009 to Mar. 2011 in 3-months time windows with 1.5 months overlaps

2.2.3 Asterisk Project

The video for the Asterisk project is also available online[3], and the results from our quantitative analysis of the betweenness centralities and the network diameters are shown in Figure 17. The results show that the network diameter remained steady at 6 throughout the period. The Asterisk community was by far the most crowded project, with 932 nodes and 4282 edges. The stacked area chart shows the distribution of centralities, where we see an 80%-20% distribution (i.e., 80% or more of the activity is attributed to six major players, with the rest of the community accounting for only 20%). This is evident in the video representation as well, as the top-level structure of the network holds throughout the time period. The results from the visual and quantitative analysis links the Asterisk fork to the more community-driven category of forking reasons.

[3]Video visualizations available at http://eecs.oregonstate.edu/~azarbaam/OSS2014/

2.2.4 Initial study conclusion

We studied the collaboration networks of three FOSS projects using a combination of temporal visualization and quantitative analysis. We based our study on two papers by Robles and Gonzalez-Barahona [46] and Hannemann and Klamma [23], and identified three projects that had forked in the recent past. We mined the collaboration data, formed dynamic collaboration graphs, and measured social network metrics over an 18-month period time window.

We also visualized the dynamic graphs (available online) and as stacked area charts over time. The visualizations and the quantitative results showed the differences among the projects in the three forking reasons of personal differences among the developer teams, technical differences (addition of new functionality) and more community-driven development. The novelty of the approach was in applying the network-specific *temporal* analysis rather than static analysis, and in the temporal visualization of community structure. We showed that this approach shed light on the structure of these projects and reveal information that cannot be seen otherwise.

More importantly, the initial study showed the limitations of a *network-specific* approach, and hence, we adopted a *population-processes* approach for our main study as explained in section 3.

3 Appendix C: Mathematical Definition of Effects in the Statistical Model

3.1 Structural Effects for the Objective Function

1. Reciprocity

$$\sum_j x_{ij}x_{ji}$$

2. Closure effect: Transitive triplets

$$\sum_{j,h} x_{ih}x_{ij}x_{jh}$$

3. Closure effect: Transitive ties

$$\sum_j x_{ij} max_h(x_{ih}x_{hj})$$

4. Closure effect: Balance

$$\sum_{j=1}^{n} x_{ij} \sum_{h=1,h\neq i,j}^{n} |(b_0 - x_{ih} - s_{jh})|$$

b_0 is a constant equal to the mean of $|x_{ih} - x_{jh}|$

5. Closure effect: The number of developers at distance two

$$\sum_j (1 - x_{ij}) max_h(x_{ih}x_{hj})$$

6. Three-cycles

$$\sum_j x_{ij} \sum_h x_{jh}x_{hi}$$

7. Betweenness

$$\sum_{j,h} x_{hj}x_{ij}(1 - x_{hj})$$

8. Density (or out-degree)

$$\sum_j x_{ij}$$

9. Activity: Out-degree

$$\sum_j x_{ij}x_{j+} = \sum_j x_{ij} \sum_h x_{jh}$$

10. Covariate V-related popularity

$$\sum_j x_{ij}v_j$$

11. Covariate V-related activity

$$v_i x_{i+}$$

12. Covariate V-related dissimilarity

$$\sum_j x_{ij}|v_i - v_j|$$

13. in-in degree assortativity

$$\sum_j x_{ij}x_{+i}^{1/c}x_{+j}^{1/c}$$

with c = 1 or 2

14. in-out degree assortativity

$$\sum_j x_{ij}x_{+i}^{1/c}x_{j+}^{1/c}$$

with c = 1 or 2

15. out-in degree assortativity

$$\sum_j x_{ij}x_{i+}^{1/c}x_{+j}^{1/c}$$

with c = 1 or 2

16. out-out degree assortativity

$$\sum_j x_{ij}x_{i+}^{1/c}x_{j+}^{1/c}$$

with c = 1 or 2

3.2 Behavior-related Effects

For a behavior variable $s_{ik}^z(x,z)$, the following formulas cab be used.

1. Shape z_i

2. Quadratic Shape z_i^2

3. Total Similarity
$$\sum_j x_{ij}(sim_{ij}^z - \bar{sim}^z)$$
where
$$sim_{ij}^z = (1 - |z_i - z_j|/max_{ij}|z_i - z_j|)$$

4. Average Similarity
$$\frac{1}{x_{i+}}\sum_j x_{ij}(sim_{ij}^z - \bar{sim}^z)$$
where
$$sim_{ij}^z = (1 - |z_i - z_j|/max_{ij}|z_i - z_j|)$$

5. Average alter
$$\frac{z_i(\sum_j x_{ij} z_j)}{\sum_j x_{ij}}$$

Bibliography

[1] Asur, S., S. Parthasarathy, and D. Ucar, (2009), "*An event-based framework for characterizing the evolutionary behavior of interaction graphs,*" ACM Trans. Knowledge Discovery Data. 3, 4, Article 16, (November 2009), 36 pages. 2009.

[2] Azarbakht, A. and C. Jensen, "*Drawing the Big Picture: Temporal Visualization of Dynamic Collaboration Graphs of OSS Software Forks,*" Proc. 10th Int'l. Conf. Open Source Systems, 2014.

[3] Azarbakht, A. and C. Jensen, "*Temporal Visualization of Dynamic Collaboration Graphs of OSS Software Forks,*" Proc. Int'l. Network for Social Network Analysis (INSNA) Sunbelt XXXIV Conf., 2014.

[4] Azarbakht, A., "*Drawing the Big Picture: Analyzing FLOSS Collaboration with Temporal Social Network Analysis,*" Proc. 9th Int'l. Symp. Open Collaboration, ACM, 2013.

[5] Azarbakht, A. and C. Jensen, "*Analyzing FOSS Collaboration & Social Dynamics with Temporal Social Networks,*" Proc. 9th Int'l. Conf. Open Source Systems Doct. Cons., 2013.

[6] Azarbakht, A., "*Temporal Visualization of Collaborative Software Development in FOSS Forks,*" Proc. IEEE Symp. Visual Languages and Human-Centric Computing, 2014.

[7] Baishakhi R., C. Wiley, and M. Kim, "*REPERTOIRE: a cross-system porting analysis tool for forked software projects,*" Proc. ACM SIGSOFT 20th Int'l. Symp. Foundations of Software Engineering, ACM, 2012.

[8] Bastian, M., S. Heymann, and M. Jacomy, "*Gephi: an open source software for exploring and manipulating networks,*" Int'l AAAI Conf. on Weblogs and Social Media, 2009.

[9] Bezrukova, K,, C. S. Spell, J. L. Perry, "*Violent Splits Or Healthy Divides? Coping With Injustice Through Faultlines,*" Personnel Psychology, Vol 63, Issue 3. 2010.

[10] Bird, C., D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "*Latent social structure in open source projects,*" Proc. 16th ACM SIGSOFT Int'l. Symposium on Foundations of software engineering, ACM, 2008.

[11] Brandes, U. "*A Faster Algorithm for Betweenness Centrality*", Journal of Mathematical Sociology 25(2):163-177, 2001.

[12] Chakrabarti, D., and C. Faloutsos. "*Graph mining: Laws, generators, and algorithms,*" ACM Computing Surveys, 38, 1, Article 2, 2006.

[13] Chen, C. and Liu, Lon-Mu, "*Joint Estimation of Model Parameters and Outlier Effects in Time Series,*" Journal of the American Statistical Association, 88, 284–297. 1993.

[14] Coleman, J.S. "*Introduction to Mathematical Sociology,*" New York etc.: The Free Press of Glencoe. 1964.

[15] Crowston, K., K. Wei, J. Howison, and A. Wiggins. "*Free/Libre open-source software development: What we know and what we do not know,*" ACM Computing Surveys, 44, 2, Article 7, 2012.

[16] Davidson, J, R. Naik, A. Mannan, A. Azarbakht, C. Jensen, "*On older adults in free/open source software: reflections of contributors and community leaders,*" Proc. IEEE Symp. Visual Languages and Human-Centric Computing, 2014.

[17] Ernst, N., S. Easterbrook, and J. Mylopoulos, "*Code forking in open-source software: a requirements perspective,*" arXiv preprint arXiv:1004.2889, 2010.

[18] Feuerriegel S. and N. Proellochs. "*SentimentAnalysis: Dictionary-based sentiment analysis*", R package version 1.1-0. https://github.com/sfeuerriegel/SentimentAnalysis. 2016.

[19] Ford, L. R. and D. R. Folkerson, "*A simple algorithm for finding maximal network flows and an application to the Hitchcock problem,*" Canadian Journal of Mathematics, vol. 9, pp. 210-218, 1957.

[20] Forrest, D., C. Jensen, N. Mohan, and J. Davidson, "*Exploring the Role of Outside Organizations in Free/ Open Source Software Projects,*" Proc. 8th Int'l. Conf. Open Source Systems, 2012.

[21] Fruchterman, T. M. J. and E. M. Reingold, "*Graph drawing by force-directed placement,*" Softw: Pract. Exper., vol. 21, no. 11, pp. 1129-1164, 1991.

[22] Guzzi, A., A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen. "*Communication in open source software development mailing lists,*" Proc. 10th Conf. on Mining Software Repositories, IEEE Press, 2013.

[23] Hannemann, A and , R. Klamma "*Community Dynamics in Open Source Software Projects: Aging and Social Reshaping,*" Proc. Int. Conf. on Open Source Systems, 2013.

[24] Heider, F. The Psychology of Interpersonal Relations. John Wiley & Sons. 1958.

[25] Howison, J. and K. Crowston. "*The perils and pitfalls of mining SourceForge,*" Proc. Int'l. Workshop on Mining Software Repositories, 2004.

[26] Howison, J., K. Inoue, and K. Crowston, "*Social dynamics of free and open source team communications,*" Proc. Int'l. Conf. Open Source Systems, 2006.

[27] Howison, J., M. Conklin, and K. Crowston, "*FLOSSmole: A collaborative repository for FLOSS research data and analyses,*" Int'l. Journal of Information Technology and Web Engineering, 1(3), 17-26. 2006.

[28] Krivitsky, P. N., and M. S. Handcock. "*A separable model for dynamic networks,*" Journal of the Royal Statistical Society: Series B (Statistical Methodology) 76, no. 1: 29-46. 2014.

[29] Kuechler, V., C. Gilbertson, and C. Jensen, "*Gender Differences in Early Free and Open Source Software Joining Process,*" Open Source Systems: Long-Term Sustainability, 2012.

[30] Kunegis, J., S. Sizov, F. Schwagereit, and D. Fay, "*Diversity dynamics in online networks,*" Proc. 23rd ACM Conf. on Hypertext and Social Media, 2012.

[31] Leskovec, J., Kleinberg, J., and Faloutsos, C.: "*Graphs over time: densification laws, shrinking diameters and possible explanations,*" Proc. SIGKDD Int'l. Conf. Knowledge Discovery and data Mining, 2005.

[32] Leskovec, J., K. J. Lang, A. Dasgupta, and M. W. Mahoney, "*Statistical properties of community structure in large social and information networks,*" Proc. 17th Int'l. Conf. World Wide Web, ACM, 2008.

[33] Lopez-de-Lacalle, J. "*tsoutliers: Detection of Outliers in Time Series*", R package version 0.6-5. https://CRAN.R-project.org/package=tsoutliers, 2016.

[34] Nakakoji, K., Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye. "*Evolution patterns of open-source software systems and communities,*" Proc. Int'l. Workshop Principles of Software Evolution, ACM, 2002.

[35] Mikkonen, T., L. Nyman, "*To Fork or Not to Fork: Fork Motivations in SourceForge Projects,*" Int'l. J. Open Source Softw. Process. 3, 3. July, 2011.

[36] Noack, A., "*Energy models for graph clustering,*" J. Graph Algorithms Appl., vol. 11, no. 2, pp. 453-480, 2007.

[37] Nowak, M. A. "*Five rules for the evolution of cooperation,*" Science 314, No. 5805: 1560-1563. 2006.

[38] Nyman, L. , "*Understanding code forking in open source software,*" Proc. 7th Int'l. Conf. Open Source Systems Doct. Cons., 2011.

[39] Nyman, L., T. Mikkonen, J. Lindman, and M. Fougère, "*Forking: the invisible hand of sustainability in open source software,*" Proc. SOS 2011: Towards Sustainable Open Source, 2011.

[40] Nyman, L., "*Hackers on Forking,*" Proc. Int'l. Symp. on Open Collaboration, 2014.

[41] Oh, W., Jeon, S., "*Membership Dynamics and Network Stability in the Open-Source Community: The Ising Perspective*" Proc. 25th Int'l. Conf. Information Systems. 2004.

[42] Page, B, B. Sergey, R. Motwani and T. Winograd, "*The PageRank Citation Ranking: Bringing Order to the Web,*" Technical Report, Stanford InfoLab, 1999.

[43] Proellochs, Feuerriegel and Neumann: "*Generating Domain-Specific Dictionaries Using Bayesian Learning*", Proceedings of the 23rd European Conference on Information Systems (ECIS 2015), Muenster, Germany, 2015.

[44] R Core Team. "*R: A language and environment for statistical computing. R Foundation for Statistical Computing*", Vienna, Austria. URL https://www.R-project.org/. 2016.

[45] Robins, G., P. Pattison, Y. Kalish, and D. Lusher. "*An introduction to exponential random graph (p*) models for social networks,*" Social networks 29, no. 2: 173-191. 2007.

[46] Robles, G. and J. M. Gonzalez-Barahona, "*A comprehensive study of software forks: Dates, reasons and outcomes,*" Proc. 8th Int'l. Conf. Open Source Systems, 2012.

[47] Rocchini, C. (Nov. 27 2012), Wikimedia Commons, Available: http://en.wikipedia.org/wiki/File:Centrality.svg, 2012.

[48] Singer, L., F. Figueira Filho, B. Cleary, C. Treude, M. Storey, and K. Schneider. "*Mutual assessment in the social programmer ecosystem: an empirical investigation of developer profile aggregators,*" Proc. Conf. Computer supported cooperative work, ACM, 2013.

[49] Snijders, T. AB. "*Markov chain Monte Carlo estimation of exponential random graph models,*" Journal of Social Structure 3, no. 2: 1-40. 2002.

[50] Snijders, Tom AB. "*Models for longitudinal network data,*" Models and methods in social network analysis 1: 215-247. 2005.

[51] Snijders, Tom AB., GG Van de Bunt, CEG Steglich, "*Introduction to stochastic actor-based models for network dynamics,*" Social networks 32 (1), 44-60. 2010.

[52] Sowe, S., L. Stamelos, and L. Angelis, "*Identifying knowledge brokers that yield software engineering knowledge in OSS projects,*" Information and Software Technology, vol. 48, pp. 1025-1033, Nov 2006.

[53] Spence, M. "*Job market signaling,*" Quarterly Journal of Economics, 87: 355-374. 1973.

[54] Steglich, C., T. AB Snijders, and M. Pearson. "*Dynamic networks and behavior: Separating selection from influence,*" Sociological methodology 40, no. 1: 329-393. 2010.

[55] Storey, M., L. Singer, B. Cleary, F. Figueira Filho, and A. Zagalsky, "*The (R) Evolution of social media in software engineering,*" Proc. Future of Software Engineering, ACM, 2014.

[56] Syeed, M. M., "*Socio-Technical Dependencies in Forked OSS Projects: Evidence from the BSD Family,*" Journal of Software 9.11 (2014): 2895-2909. 2014.

[57] Teixeira, J., and T. Lin, "*Collaboration in the open-source arena: the webkit case,*" Proc. 52nd ACM conf. Computers and people research (SIGSIM-CPR '14). ACM, 2014.

[58] Torres, M. R. M., S. L. Toral, M. Perales, and F. Barrero, "*Analysis of the Core Team Role in Open Source Communities,*" Int. Conf. on Complex, Intelligent and Software Intensive Systems, IEEE, 2011.

[59] Zachary, W., "*An information flow model for conflict and fission in small groups,*" Journal of Anthropological Research, vol. 33, no. 4, pp. 452-473, 1977.