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

Prelim Report

Work Under Supervision of Prof. Carlos Jensen
Ph.D. Committee Members:
Prof. Alex Groce
Prof. Chris Scaffidi
Prof. Drew Gerkey

November 10, 2015

1 Abstract

Social networks 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 group are formed around communities of interest, and depend on continued interest and involvement. Forking, either as an acrimonious split or a friendly divide, affects the community. Most existing research on forking is post-hoc. The run-up to fork is seldom studied. This leaves several questions unanswered. In this study, we propose using statistical modeling of longitudinal social collaboration graphs of software developers to study the evolution and social dynamics of FOSS communities. With these techniques 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. One set of dynamics we are especially interested in, are those that lead FOSS projects to fork. This will help predict formation of unhealthy dynamics, which 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 [50]. 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 [35].

Community splits in free and open source software development are referred to as forks, and are relatively common. Robles et al. [41] 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. [41] 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), to the study by Baishakhi et al. [7] on postforking porting of new features or bug fixes from peer projects. It encompasses works of Nyman on developers' opinions about forking [37], developers motivations for performing forks [32], the necessity of code forking as tool for sustainability [36], and Syeed's work on sociotechnical dependencies in the BSD projects family [51].

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. 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 [18]. Oh et al. [38] 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 unconstructive 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 OSS 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" [38]. Identification of recipes for success or stagnation, sustainability or fragmentation may lead to a set of best practices and pitfalls.

Our research proposes using 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 [41]

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

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

Figure 1: The frequency of reasons for forking as classified by Robles and Gonzalez-Barahona [41]

Main Reasons for Forking
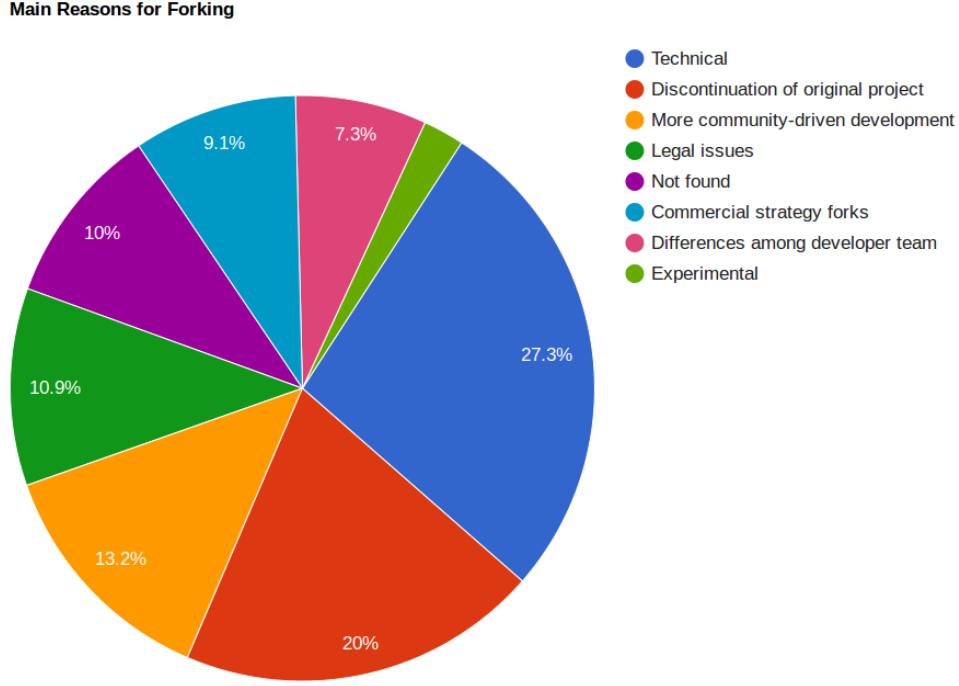
research objective and research questions. After that, in section 5 presents how data gathering, statistical modeling, and qualitative analysis are proposed to be done, as well as our initial hypotheses. In section 6, we present results of our initial study. Section 8, shows the proposed timeline for the research and the author's graduation. And lastly, section 9 we discuss the threats to validity.

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][20][24][25][31], identifying knowledge brokers and associated activities [47], project sustainability [31][36], forking [35], requirement satisfation [16], their topology [10], their demographic diversity [28], gender differences in the process of joining them [27], and the role of

age and the core team in their communities [2][3][15][53]. Most of these studies have tended to look at community as a static structure rather than a dynamic process [14]. 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 codebase) 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. [52] They found that *coopetition* (both competition and collaboration) exists in the open source community; morover, 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" [52].

The study of communities has grown in popularity in part thanks to advances in social network analysis. From the earliest works by Zachary [54] to the more recent works of Leskovec et al. [29][30], 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 [54]. 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 [17] 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.[54]

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 2: 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. [24]. 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.

The tension between diversity and homogeneity in a community was studied by Kunegis et al. [28]. They defined five network statistics, listed in Table 3, 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. [28] 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 [21] on three open source bioinformatics communities. They measured "age" of users, as starting from their first activity and found

Table 3: The measures of diversity used by Kunegis et al. [28]

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

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 reflects the changes the community goes through. And so, it can be used to describe the context surrounding a forking event. At least, three of the six main reasons for forking [41], as listed in Table 1 are socially related, and so, their effects should be reflected in the developers' interactions data. For example, if a fork occurred because of a desire for "more community-driven development", we expect to see interaction patterns in the collaboration data showing a strongly-connected core that is hard to penetrate for the rest of the community. In other words, in this case, the power stays in the hands of the same people throughout, as new

people come and go.

This study plans 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 research objective and research questions are listed in the following subsection 4.1.

4.1 Research Objective:

What are the social patterns associated with different types of undesirable forking?

4.1.1 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.

4.1.2 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.

4.1.3 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.

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.

5.1   Phase 1: Data Collection

5.1.1   Not all forks are equal: Undesirable forking

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

Table 4: Not all forks are bad; The three types of projects for which data is collected

Type of forking	Abbreviation
Undesirable forking	U.F.
Other (Healthy/Other) forking	H.F.
No forking at all	No.F.

Our results will only be valid if the patterns found in the *undesirable forking* (U.F.) category are unique to that category, and is not shared in the other two control groups (H.F. and No.F.).

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 [41]. They found the reasons behind each fork, as listed in 1. We applied three selection criteria to the 220 projects on that list to find projects in U.F. and H.F. categories. A project was short-listed as a U.F. or H.F. if **a)** it was recent, i.e., happened after the year 2000, **b)** its data was accessible 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; large enough to make a sociogram for our statistical analysis. For No.F. category, we chose well-known, stable projects, that, have been active long enough (to experience turbulence); and yet have not forked; and are similar in size of the development team.

Table 5: Projects in U.F. H.F., and No.F. categories 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.
ApacheCouchDB & 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	Not forked	No.F.
Python	Not forked	Not forked	No.F.
OpenStack Neutron	Not forked	Not forked	No.F.
GlusterFS	Not forked	Not forked	No.F.

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 5.

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. **d)** (Proposed) The

Table 6: All projects forked because of *"personal differences among the developer team"* (U.F.) [41] in chronological order. U.A. means Unavailable

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

content of the messages send and received by the top contributors of the project in the months leading to the forking events can be a good data source to be sentiment-analyzed. Using sentiment analysis, we can quantify the level of positive/negative language usage, and may detect events, for example, in-fighting where people involved in the conflict exchange negative sentiments.

The time period for which data was collected is the year in which 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 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; **a)** a *cross-sectional* study, and **b)** a *longitudinal* study. We are interested in patterns in the run-up to forks, so, we choose to do a longitudinal study, unlike most existing research on forking. This is a novel contribution of our study.

A longitudinal study can look at the sociograms in two distinct approaches: **a)** a *network-specific* approach, in which, we focus on the actual networks under study, to describe the structure of the *observed networks* with numeric summaries/descriptors to describe the structure and what we actually see, e.g., in Figures 3 and 4. Or, **b)** a *population-processes* approach, in which, we treat the observed network as one realization from a set of all possible networks with the same number of nodes, and with similar characteristics. To this end, the observed network, is only good to help us understand the social forces/processes that generated it.

Figures 3 and 4 are good examples of the expressiveness as well as the limitation of a network-specific approach. Figure 3 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 4 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.

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 need to find a well-fitting statistical model of our observed interactions network, because of the following reasons:

5.2.1.1 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.

5.2.1.2   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 [22].

5.2.1.3   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 [40].

5.2.1.4   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 [40].

5.2.1.5   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 [40].

5.2.1.6   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.)[40]

5.2.2 Theories

Software development is a collective effort, and software developers are human beings, so, theories about human behavior from sociology and psychology could help explain how software developers interact and collectively develop software. In particular, the following three theories about human behavior are the bases for the hypotheses in this research study. They are complement each other in term of explaining the context of forking .

5.2.2.1 Balance theory [22] 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.

5.2.2.2 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.

5.3 Statistical Model

Longitudinal evolution of a network data is the result of many small atomic changes occurring between the the consecutively observed networks. In our case, software developers are the actors in the networks, and they can form a connection with another actor, break off an existing connection, or maintain their status quo. These are the four possibilities of atomic change within our evolving networks. (status quo counts twice: maintaining a non-connection or 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* [46], based on a model of actors 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.

Let the data for our statistical actor-oriented model be M repeated observations on a network with g actors. 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 [13].

For each change, the model focuses on the actor whose tie is changing. We assume that actor i has control over the set of outgoing tie variables $(X_{i1}, ..., X_{ig})$ (i.e. the ith 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 actor i changes of 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*. These three functions and their definitions taken from [45] are explained in detail the following subsections.

5.3.1 Rate Function

The *rate function* $\lambda_i(x)$ for actor i is the rate at which in actor i's outgoing ties changes occur. It models how frequently the actors make ministeps. The rate function is formally defined [45] by

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

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

5.3.2 Objective Function

The *objective function* $f_i(s)$ for actor 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 $(X_{i1}, ..., X_{ig})$, actor i selects the change that gives the greatest increase in the objective function. We assume that if there is difference between actors in their objective functions, these differences can be represented based on the model covariates [45].

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 actor 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 actor i chooses to change his tie variable with the other actor j for whom the following value [45] is the highest:

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

This is a myopic stochastic optimization rule: myopic 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 actor i chooses to change x_{ij} for any particular j, given that such a change happens, is [45],

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

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 actor i is the value attached to this actor (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 actor i chooses to change x_{ij} for that other actor j for whom the following value [45] 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 actor-oriented model, described above, define a continuous-time Markov chain on the space χ of all directed graphs on this set of g actors.

The transition rate matrix (also called intensity matrix), for this model is given by

$$q_{ij}(x) = \lim_{dt \to 0} \frac{1}{dt} P\big(X(t + dt) = X(i \mapsto j) | X(t) = x\big)$$

$$= \lambda_i(x) p_{ij}(x) \quad (6)$$

Expression (6) shows the rate at which actor 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 [45]:
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 actor i is chosen randomly with probabilities $\frac{\lambda_i(x)}{\lambda_+(x)}$

3. Given this i, choose actor 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

The functions described above need to be supplied 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. In the following, we explain what functions can be included in these functions.

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 [45]:

5.4.1.1 Structural Effects

1. The reciprocity effect

2. The closure effect

   For example,

   (a) Transitive triplets effect.
   (b) Transitive ties.
   (c) Balance
   (d) Number of actors at distance two

3. Three-cycles

4. Betweenness represents brokerage

5. Density

6. Activity

7. Actor-oriented covariate V-related popularity

8. Actor-oriented covariate V-related activity

18

9. Actor-oriented covariate V-related dissimilarity

10. in-in degree assortativity

11. in-out degree assortativity

12. out-in degree assortativity

13. out-out degree assortativity

5.4.1.2 Behavior-related Effects

Let's denote a dependent behavior variable as Z.

1. Shape effect: Explains the drive toward high values on Z

2. Quadratic Shape Effect

3. Similarity

   (a) average similarity
   (b) total similarity

4. average alter effect

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 [45]:

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:
   $r_{ij3}(x) = x_{ij}w_{ij}$

5.5   Markov Chain Monte Carlo (MCMC) Estimation

Network evolution can be simulated using a MCMC estimation method. The method of moments (MoM) can be used in the following way [45]:

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 [45]:

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 $\hat{\beta}$ for which the expected and the observed vectors are the same.

5.6 Hypothesis Testing

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

5.7.1 Null Hypothesis # 1:

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

5.7.2 Null Hypothesis # 2:

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

5.7.3 Null Hypothesis # 3:

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

6 Initial study: Temporal network-specific analysis

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 6, 7, and 8 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
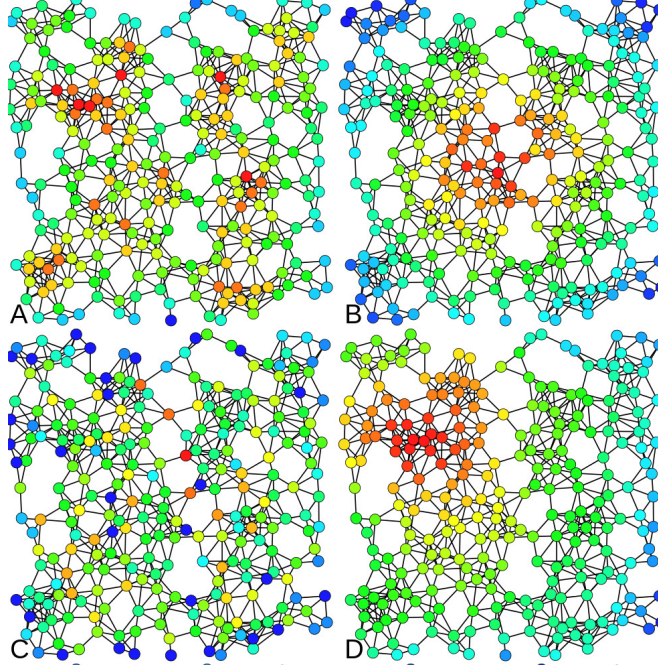
Figure 2: Heat-map color-coded examples are shown above. Nodes with higher centrality metric are colored with warmer color: red is the warmest color here. The same network is analyzed four times with the following centrality measures: A) Degree centrality, B) Closeness centrality, C) Betweenness centrality and D) Eigenvector centrality [42]

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, 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 [39] 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 *re-*

silience 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].

6.1 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 [19] layout and Noack's LinLog [33].

6.2 Initial study results and discussion

6.2.1 Kamailio Project

Figure 5 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

---

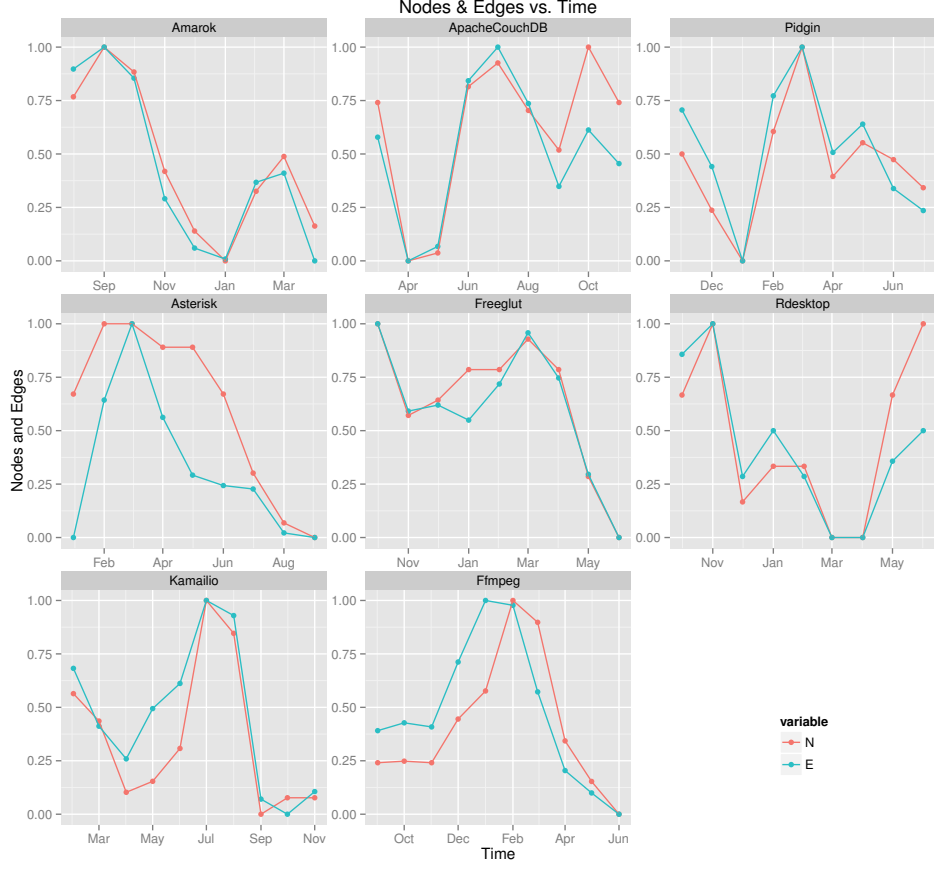[1]Video visualizations available at http://eecs.oregonstate.edu/~azarbaam/OSS2014/

Figure 3: 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.

retirement. This corresponds to the personal difference category of forking reasons.

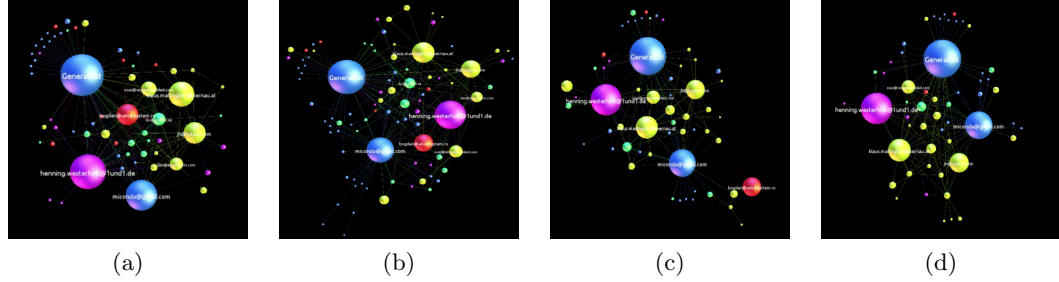Figure 6 shows the betweenness centrality of the major contributors of

Figure 4: 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.

Kamailio project over the same time period. The horizontal axis marks the dates, (each mark represents a 3-month time window with 1.5 months overlap). 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

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

Figure 5: 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.
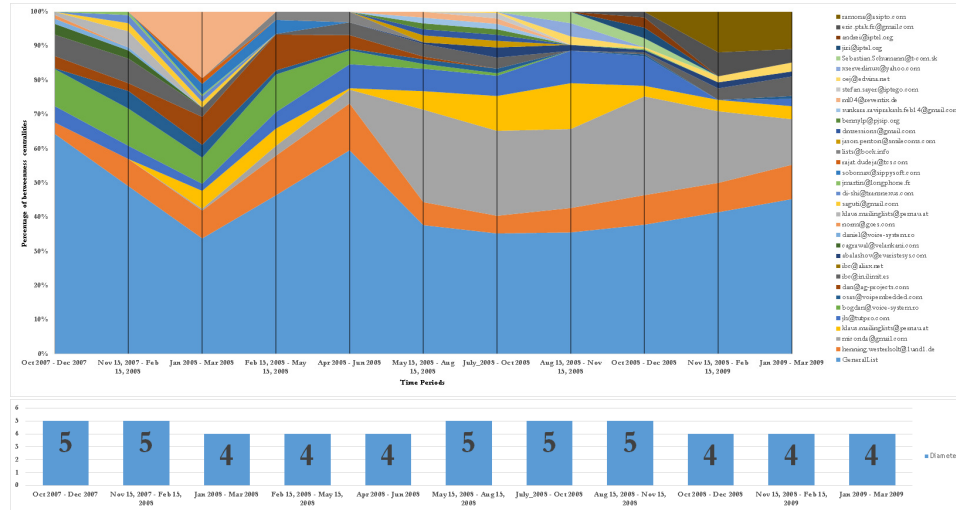

Figure 6: Kamailio top contributors' betweenness centralities and network diameter over time (Oct. 2007 to Mar. 2009) in 3-month time windows with 1.5-month overlaps

and in 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 6. An increase in the network diameter during this period is noticeable; this coincides with findings of Hannemann and Klamma [21].
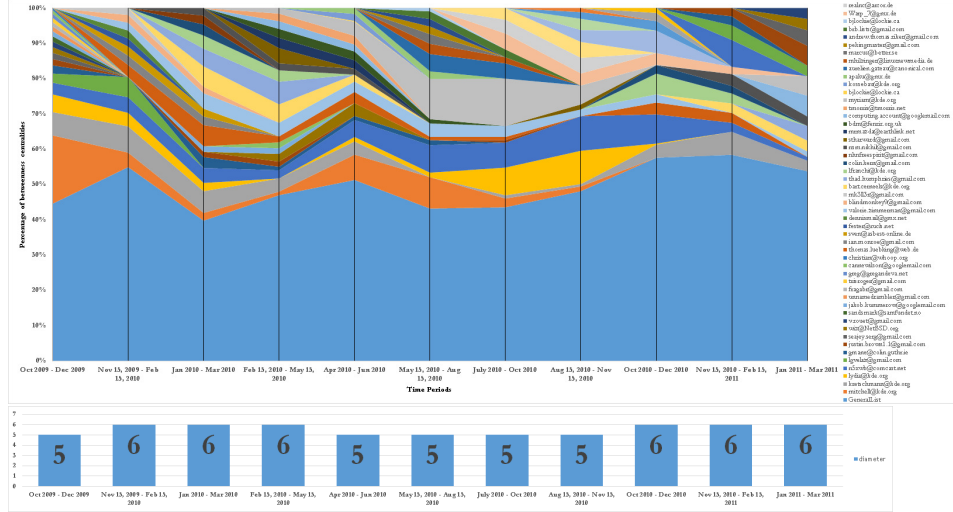
27

Figure 7: 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

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.

6.2.2 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 7. 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 healthy 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.
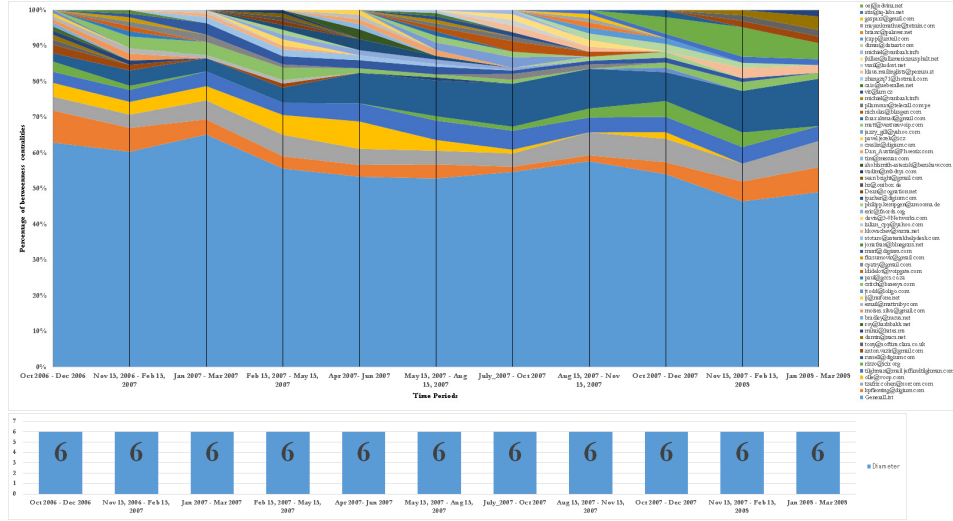
28

Figure 8: 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

6.2.3 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 8. 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.

6.2.4 Initial study conclusion

We studied the collaboration networks of three FOSS projects using a combination of temporal visualization and quantitative analysis. We based our

---

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

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

study on two papers by Robles and Gonzalez-Barahona [41] and Hannemann and Klamma [21], 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 need to adopt a *population-processes* approach for our proposed main study.

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 conduct the interview study, 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.

8 Timeline

Table 9 outlines my proposed research timeline. It also shows the mapping between research questions and the phases of the study.

TABLE 9   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 2015	Planning and preliminary examination
Fall 2015	Prelim
Winter 2015	Data collection for issue tracking and source code
Winter/Spring 2015	Statistical study, longitidinal modeling RQ 1-3
Spring 2016	Thesis writing
June 2016	Defense (Hopefully)

9   Threats to Validity

The presented 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, partly because of the availability of their data. Given access, a larger number of projects as the sample size could result in a more robust investigation. Second, the proposed technique uses the 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.

Acknowledgment

Table 8: Examples of explanatory variables to include in the model. [49] The network is denoted by x, where x_{ij} stands for the value of the directed relationship between actors i and j. The behavioral variable is denoted by z. We assume that $x_{ij} = 1$ stands for presence of a tie and $x_{ij} = 0$ for absence.

Network effect	Network Statistic	Description
Outdegree	$\sum_j x_{ij}$	Overall tendency to have ties (Negative parameter means, on average, cost of friendship ties higher than their benefits)
Reciprocity	$\sum_j x_{ij}x_{ji}$	Tendency to have reciprocated ties
Balance	$\sum_j x_{ij} strsim_{ij}$	Tendency to have ties to structurally similar others (structural equivalence with respect to outgoing ties)
Covariate similarity	$\sum_j x_{ij} sim_{ij}$	Tendency to have ties to similar others (homophile selection)
3-cycles	$\sum_j x_{ij} \sum_h x_{jh}x_{hi}$	Tendency to form relationship cycles (negative parameter means absense of hierarchy)
Transitive triplets	$\sum_j x_{ij} \sum_h x_{ih}x_{hj}$	Tendency toward triadic closure of the neighborhood (linear effect of the number of indirect ties)
Transitive ties	$\sum_j x_{ij} max_h(x_{ih}x_{hj})$	Tendency toward triadic closure of the neighborhood (binary effect of indirect ties)
Covariate alter	$\sum_j x_{ij}(z_j - \bar{z})$	Main effect of alter's behavior (covariate determines popularity in network)
Actors at distance 2	$\sum_j (1-x_{ij}) max_h(x_{ih}x_{hj})$	Tendency to keep others at social distance 2 (negative measure of triadic closure; lower means stronger network closure)

33

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

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

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

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

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

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

[36] 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.

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

[38] 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.

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

[40] 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.

[41] 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.

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

[43] 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.

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

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

[46] 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.

[47] 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.

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

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

[50] 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.

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

[52] 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.

[53] 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.

[54] 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.