

Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

BJÖRN MEIER
MASTER THESIS

**COLLABORATION NETWORKS
IN OPEN SOURCE PROJECTS'
ISSUE TRACKERS**

Submitted on 11 November 2015

Supervisor: Prof. Dr. Dirk Riehle, M.B.A.
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 11 November 2015

License

This work is licensed under the Creative Commons Attribution 4.0 International license (CC BY 4.0), see <https://creativecommons.org/licenses/by/4.0/>

Erlangen, 11 November 2015

Abstract

Open source software nowadays is increasingly used outside its own ecosystem. It is used by governmental departments and companies, for commercial projects and as a part of critical infrastructure, like encryption libraries. This usually makes it necessary to increase the collaboration with open source communities but it is also required to asses new risks, which arise from using a software, maintained mainly by volunteers. In order to achieve this we need a better understanding of open source communities and about how they organize and structure themselves, whether they depend on key personality and whether they form subcommunities? How do such communities change their behavior over time, during growth and which is the most efficient form of structuring to handle reported issues?

We collect issue tracker data and use it to create sequences of social networks for 6007 projects present on Github.com and for 120 projects from the Apache Software Foundation. Based on metrics to quantify the strength of subcommunities and centralization we study the general structure of open source communities and how they might correlate, but also their behavior over time. We compare the communities based on an efficiency metric to gain information about preferable structures.

Our results show that the most open source communities avoid to organize themselves in subcommunities while they are highly dependent on a few key personalities. The results of both metrics do indeed correlate, which means that if a community has strongly distinct groups it is unlikely to be highly centralized. But neither the few in subcommunities organized projects nor any other organizational type show a significantly higher efficiency. Although projects grow over time, they show only little internal structural change.

The stability of open source communities and the strong avoidance of subcommunities are unexpected results, since it highly contradicts recent related studies and therefore requires more research to better predict the development of projects. For the assessment of projects the dependency on strong key members for the stability may prove helpful when it comes to indicating and analyzing major changes.

Contents

1	Introduction	1
2	Research	2
2.1	Introduction	2
2.2	Related work	3
2.3	Research questions	4
2.4	Research approach	6
2.4.1	Social networks	6
2.4.2	Structural metrics	9
2.4.3	Validation of network properties	11
2.4.4	Relationships	13
2.4.5	Efficiency	13
2.5	Used data sources	14
2.6	Research results	15
2.6.1	Centralization but no modularization	15
2.6.2	From centralization to modularization	16
2.6.3	Property relationships	18
2.6.4	The development of projects	19
2.6.5	Structures and their efficiency	21
2.7	Discussion	22
2.8	Conclusion	23
	Appendices	25
Appendix A	Scatter plots for paired network properties	25
Appendix B	Highly centralized and clustered networks	32
Appendix C	Temporal correlations for modularity, centralization and size	33
Appendix D	Modularity and centralization density kernels based on efficiency	36
Appendix E	Modularity and centralization distributions depending on project age	37

1 Introduction

Thesis description

We have an existing framework for crawling data from Github.com issue trackers. Objective of this thesis is to extend the framework to also crawl Jira issue trackers and then perform an analysis of the evolution of selected metrics from the field of social network analysis based on a representative set of Github.com and Apache projects. The thesis presents conclusions about the behavior of collaboration in open source issue trackers and about evolutionary patterns of project metrics and networks.

2 Research

2.1 Introduction

How important Open Source Software (OSS) has become was demonstrated as bugs in two open source libraries caused major security issues accompanied by extensive media coverage in 2014. The OpenSSL cryptography library responsible for the “Heartbleed” bug¹ and the Unix-Shell Bash causing “Shellshock”², are both usually part of a mainly open source software stack to power a webserver. Thereby we are now long beyond the point where OSS was primarily used by the communities themselves, which created the software. These communities have been a part of many studies including research about the collaboration behavior inside OSS communities. Based on collaboration networks and social network analysis (SNA) selected OSS project communities were analyzed regarding their self-organization. For these networks multiple different data sources served as basis: mailing lists, revision control systems and issue tracker systems. The studies then analyzed the organizational structure either regarding centralization or regarding modularity, how distinct subcommunities are being formed.

Our work regards three points to extend or verify related work. Firstly the amount of projects analyzed in these studies, which were mostly less than ten or in the low hundreds. To get more representative data we use data collected for 6007 projects present on Github.com and 120 projects which are supervised by the Apache Foundation and hosted on their Jira issue tracking system. Secondly the combination of centralization and modularity. Studies found strong centralization in OSS communities while some studies showed the formation of strongly separated groups. We look at both to verify if both phenomena can occur simultaneously and how their relationship might be. Lastly the temporal aspect concerning the evolution of structures in communities. We study how fast and how strong projects change their organizational structures between different lifespans.

¹Common Vulnerabilities and Exposures registration: CVE-2014-0160

²Common Vulnerabilities and Exposures registration: CVE-2014-6271

For our research as well as for others there is the limitation to one system providing the collaboration data. This limits our OSS communities to people who do active commenting in issue tracking systems. Activities in mailing lists or source code editing is not considered as it would require to associate accounts of multiple systems, which belong to the same person. Our research only uses relationships extracted from conversations made in issue comments. Thereby these communities should not be considered complete but they represent a substantial part.

2.2 Related work

OSS communities have been studied on different levels over the last years. Besides collaboration in revision control systems and interactions in mailing lists issue tracking systems have been getting more and more attention and are used to analyze interaction behavior and collaboration. Crowston, Wei, Li and Howison (2006) also showed that official developer lists might not show a realistic picture of a project's team, therefore additional data sources should be acquired and analyzed to get a more accurate result. Sureka, Goyal and Rastogi (2011) surveyed work in this field of research, which includes among other things studies to solve the problem presented by Crowston et al. (2006). They confirmed the usefulness of SNA to investigate collaboration and showed that it could be used in the field of risk analysis.

Most studies which approached the general structure of OSS communities based on SNA can be generally separated in three areas: centralization, the formation of subcommunities and network topologies. Our work will combine the first two areas, which have been studied separately so far. We think a joint consideration will show a more complete picture and also if both phenomena are related.

Long and Siau (2007) explored network topologies for three projects based on extracted issue tracking data. Their findings relating to core developers are similar to Howison, Inoue and Crowston (2006). They also present empirical evidence that the network topologies for the three networks evolve from single hubs to core periphery networks.

The SNA research field concerning centralization was examined by Crowston and Howison (2005). They examined 120 SourceForge projects on centralization characteristics and found that those projects vary widely in their communication centralization. We will use a similar centralization approach but a different centrality metric for our networks. Their work was extended by Howison et al. (2006) with a focus on the change of core contributors. They found that the majority of projects keep key individuals over time and that the size of a project

is irrelevant to the rate at which those members are replaced.

Bird, Pattison, D’Souza, Filkov and Devanbu (2008) tackled the detection of subcommunities in social networks. They studied five big OSS projects by using social networks, created based on email data. They showed that this kind of data could be used to detect groups inside a community. They used a modularity based method to detect subcommunities and to quantify the separation. The quantification, based on modularity, will be the basis metric for this kind of structural measurement in our work as well. Therefore parts of our result are comparable to Crowston and Howison (2005), Howison et al. (2006) and Bird et al. (2008).

Because the community structure of OSS projects is mostly explored for specifically selected projects or only for a few characteristics in a slightly larger number in related work, our work uses a larger data set to explore the OSS communities and tries to find correlations between different structural characteristics.

The influence of Github.com for OSS has increased over the last few years and it is now one of the major OSS platforms for source code and issue reports. Consequently Github.com itself and the available data have become an important source for research. Gousios (2013) and Gousios, Vasilescu, Serebrenik and Zaidman (2014) presented a platform, which collects data from public Github.com projects and made them available. In the “Used data sources” section of our work we will explain how and why we used their data. Thung, Bissyandé, Lo and Jiang (2013) and Yu, Yin, Wang and Wang (2014) explored Github.com itself on a larger scale using, among other things, social network analysis to find influential developers or network patterns beyond project borders for a better understanding of Github.com.

2.3 Research questions

We want to make statements about community structures in OSS projects mainly by analyzing the social network properties, centralization and modularity. There are relative statements about a behavior: e.g. a community avoids or seeks to form subcommunities and there are quantifying statements about how this behavior can be numbered. To do both, we need points of comparison: the centralization and modularity of random networks. This results in our first two hypotheses:

Hypothesis 1a *The centralization distribution for collaboration networks is significantly different from the distribution for random networks.*

Hypothesis 1b *The modularity distribution for collaboration networks is significantly different from the distribution for random networks.*

Significance in this case means that the population from which random networks are sampled is statistically different from the populations, from which our centralization and modularity results originate.

If we expect that issue tracking communities show a similar behavior as the mailing list, we might assume, based on the work of Bird et al., 2008, that the gained social networks will show significant modularity values, therefore a strong tendency to form groups. However, the results by Crowston and Howison, 2005 based on SourceForge data, showed that we could expect networks with people organized around central leaders. Both do not exclude each other, which we show in section 2.6.2, but it seems unlikely that mostly self organized communities create such specific structures, leading to our next hypothesis:

Hypothesis 2 *Communication networks based on OSS issue tracker data show either a high centralization or a high modularity.*

The size of an society is an important factor for different communities, like companies, associations or other societies where people collaborate. It is not unusual that societies whose size increases or decreases change their structure, e.g. companies found departments. The question which arises is if OSS communities behave similarly. We approach this question from two sides. The first is the general relationship, meaning the bigger a network is the more central or modular it will be:

Hypothesis 3a *The modularity of issue tracker collaboration networks increases with the network size.*

Hypothesis 3b *The centralization of issue tracker collaboration networks increases with the network size.*

The second is a temporal relationship between the changes of size and the changes of modularity/centralization. Meaning, if more people join to work on a project over time the more distinct the groups or the more important central people become:

Hypothesis 4a *A project's successive change of size correlates temporally with the change of modularity.*

Hypothesis 4b *A project's successive change of size correlates temporally with the change of centralization.*

Another temporal aspect is how projects evolve, meaning if and how their structural organization changes over time leaving the network size out of consideration. There are ways to argue how projects are started and develop. For instance, OSS projects could get initiated by a group of people working directly with each other. As the project promotes and sophisticates, membership roles could get more sophisticated too and evolving leaders induce a stronger centralization. The alternative could be that as new software parts and modules are included and the software grows in volume and becomes more complex, developers split into groups, working on parts of the project. This leads to a research question we

want to answer:

Research question 1 *Do collaboration networks show higher modularity or centralization values depending on their age?*

Besides the questions about general properties and changes over time, we may ask if we could relate efficiency to specific ranges of modularity/centralization values, which could help to improve projects. In our case it is of interest if we could connect communities, which have a good way to handle new issues and resolve them without building a backlog, to specific structural properties.

Research question 2 *Which organizational structures of collaboration networks lead to efficient issue processing?*

2.4 Research approach

2.4.1 Social networks

Creating social networks

One of our aims is to have more representative results by using more extensive data from different issue tracking sources. In order to be able to compare and work on data from different sources, we used the uniform data model presented by Capraro (2013). They presented how the different issue tracking data from Github.com or systems based on Jira and Bugzilla could be unified to be used for joint analyses. We will focus on issues associated with projects, including their comment activity and the normalized issue state changes. The normalized issue state changes are the events associated with a time stamp when an issue is opened, closed or reopened.

We want to analyze the collaboration between participants in the OSS issue tracking communities. To determine who communicates with whom we use comments (including the initial description) as a main indicator, because they serve as a communication platform for bugs, new features, thoughts and other knowledge transfers. Consequently people who are commenting on the same issue are communicating to each other, leading to our social network: people are nodes and if they comment on the same issue an edge is created.

The information value of a complete network, created by all communication events (comments) would be minimal, especially the older projects become. The reason for that is that people who have never worked for the same project at the same time would appear in the same network and based on SNA we would take their relationship to one another into account. This would be a false conclusion as

the two individuals never collaborated in reality. To prevent this and to allow us to examine changes over time, we will look at a sequence of networks per project. To gain a sequence of networks for each project we must divide the communication events based on a sequence of time frames, so that we can create networks for every time frame.

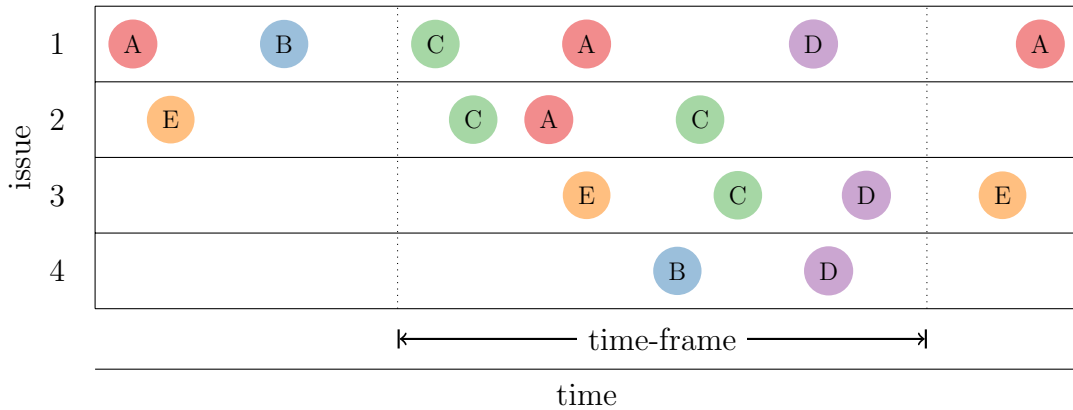


Figure 2.1: comment selection for a project's time frame based on multiple issues

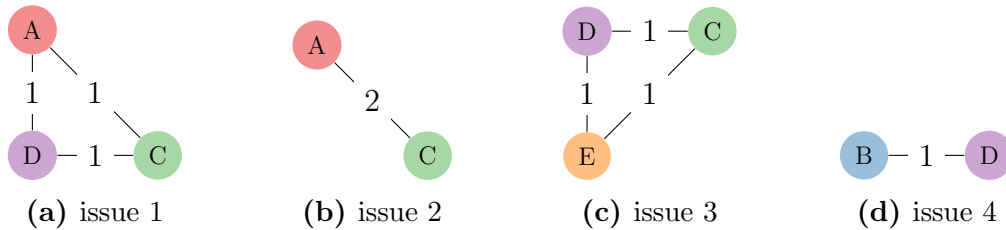


Figure 2.2: networks based on one specific issue and the activity during the time frame, see figure 2.1

We create networks for defined time frames based on comments which have been updated or created during a time frame. Figure 2.1 illustrates our method and shows which comments would be selected, namely those comments, which have been posted exactly inside a specific time-frame. It could be discussed to include the complete comment history for an issue if one comment triggers the issue during a time frame. The reason for this is that comments are likely refer to relevant comments outside the time frame. We reject this approach because the disadvantages of including inactive people during a time frame is stronger than than the gained accuracy of a complete conversation.

So far we have defined which nodes are connected but not the strength of their connection, the weight of the connecting edge. The weight is determined by the amount of comments per issue and the number of issues which a person has been

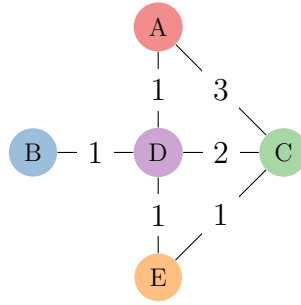


Figure 2.3: complete network for the time frame and for all issues

participating. If the same nodes are connected in multiple issue networks, the edge weights are summed up. The Figure 2.2 presents the resulting networks per issue and the combination of these issue networks results in the complete network (Figure 2.3).

Time frame size

We use a quarterly time frame to create our social networks. We also used monthly time frames to compare them to the quarterly results. In most cases the results are similar but as monthly networks have a smaller average size and we only use networks with a minimum of 10 nodes we would have to remove more networks and thus more information. This is noticeable during the development analysis as more gaps, caused by removed small networks, appear. Sometimes the monthly results also show a few spikes, which have been canceled out in our quarterly results.

Edge filtering

A big problem, when working with issue tracker data are one-time users. Capraro, 2013 determined that in their analyzed Github.com data most projects show that 40 to 70 per cent of all users are only active for one issue. As we are interested in active contributors, who do more than posting on one issue, and how they are communicating with each other and how they structure their community, we filter our networks to reduce noise. In order to do so, we remove edges from created social networks based on their edge weight. We define that the minimal recognizable interaction amount between two people has to be two. In other words, two people are only connected if both commented together at least twice on one or more issues. This is not a very restrictive limitation for interactions between people, especially for a long time period (three months), but it is already very efficient for filtering one-time and inactive users.

2.4.2 Structural metrics

To quantify structures in our social networks we will mainly use two metrics, modularity and centralization. Both values allow us to compare different networks based on their structure.

Modularity

Modularity is mainly used as a structural measurement which says how strong a network can be divided into groups/clusters. It cannot be calculated for a network directly, it can only describe the strength of connections between clusters for a provided network clustering. The modularity value can be different for the same network dependent on the clustering. Therefore the highest possible modularity for a given network is considered the modularity network property. It was introduced by Newman and Girvan, 2004 to find subcommunities in networks. The Modularity Q is defined as (Bird et al., 2008, p. 5):

$$Q = \sum_{i=1} (e_{ii} - a_i^2)$$

e_{ij} is the fraction of edges with one end's node in community i and the other in community j

a_i is the fraction of ends of edges that are attached to nodes in community i

This formula is the transformed representation of the original description, where it is described as the difference of the fraction of edges between groups and the same fraction for a degree keeping randomly connected network.

As it is not an actual calculable property like average degree or density, algorithms to find a clustering are needed. However Brandes et al., 2006 showed that finding a clustering with the maximum modularity is NP-complete. Consequently there are only optimization algorithms and heuristics providing a possibly not optimal solution. We use the very common Louvain method, a heuristic modularity optimization method published by Blondel, Guillaume, Lambiotte and Lefebvre, 2008. The value range for modularity is from zero to one.

Centrality and Centralization

Centralization measures the opposite, it says how tightly the graph is organized around its most central nodes and can be used to compare networks. Centrality refers to a value connected with a node and describes the importance of this node inside a network. If the centrality for single nodes is high and for the

remaining nodes small, the centralization value is much higher than in cases of mostly equally distributed node centralities.

There are multiple common centrality methods used in social network analysis. We use the Eigenvector centrality because it scores nodes based on their connections to influential people; therefore not only the pure value of outgoing edges count, but it also considers whom they lead to. This is a more accurate representation for central people in a OSS community than as e.g. a Degree, Betweenness or Closeness centrality.

The Eigenvector centrality is calculated by this equation:

$$Mx = \lambda x$$

The centrality c_{s_i} for node i of a network with n nodes is received from the eigenvector x belonging to the biggest eigenvalue λ . The entries are normed as follows:

$$c_{s_i} = \frac{x_i}{\|x\|_2}; 1 \leq i \leq n$$

However there is an Eigenvector centrality drawback, which still exists at this point: if disconnected subnetworks exist in a network, the Eigenvector centrality only provides centrality values for the biggest disconnected subnetwork. To solve this problem, we do the Eigenvector centrality calculation for all subnetworks separately. The resulting centrality vectors, respectively the node centralities, for each subnetwork are weighted by the fraction of number of nodes in a subnetwork:

$$c_i = c_{s_i} \frac{m(s)}{n}$$

$m(s) := \text{size of subnetwork } s$

With the node centrality values we can calculate the general network centralization as it was defined by Freeman, 1979:

$$C = \frac{\sum_{i=1}^n \max(c) - c_i}{C_{max}}$$

$$C_{max} = \sum_{i=1}^n \max(c^*) - c_i^* \text{ for the maximal centralized network of size } n$$

For Eigenvector centrality the maximal centralized network is a network of only two connected nodes and all other nodes without any connection:

$$C_{max} = \frac{n-2}{\sqrt{2}}$$

This results in our centralization value ranging from zero to one:

$$C = \frac{\sqrt{2}}{n-2} * \sum_{i=1}^n \max(c) - c_i$$

2.4.3 Validation of network properties

To decide if our OSS communities in general show significant signs of formed structures, we have to analyze the network properties. We do this by comparing centralization and modularity of the actual networks with randomly generated networks as it was done by Bird et al. (2008).

Random networks

The two basic random graph models are Erdős-Rényi's $G(n, M)$ and Gilbert's $G\{n, P(edge) = p\}$ (Bollobás, 1998). But usually, a common approach to validate discovered network structures is the generation of a control group by a degree keeping randomization (Molloy & Reed, 1995). Based on this approach Bird et al. (2008) created their comparison random networks on a link switching algorithm, which conserves a network's degree sequence, so that a node's activity is kept but who is connected to whom is shuffled. This way you keep a degree of similarity to the original network but you still get a random control group by generating thousands of random networks for every network you have. Due to the following limitations, randomization by link switching generates a distorted control group for our networks.

In Figure 2.4 it can be seen that most networks have 10 to 25 nodes. The bottom limit is created by our regulation only to use networks with a minimum of 10 nodes. The possible high densities and the resulting high centralization values, which should be validated, turn out to be problematic for a degree keeping randomization.

The link switching approach results in a significant amount of networks, where link switching is not possible or only a few isomorphic networks exist. Figure 2.6a is an example network with 15 nodes and a typical centralization of 0.58 and the Figures 2.6b and 2.6c are the only two possible networks, which are not isomorphic to the original one or to themselves. Normally a network would be randomized several thousands of times by link switching and would be added to the control group. If we do this with networks like the example network we would bias the random control group, because of adding the same networks multiple times or by not adding the correct amount of randomized networks. To avoid this problem we

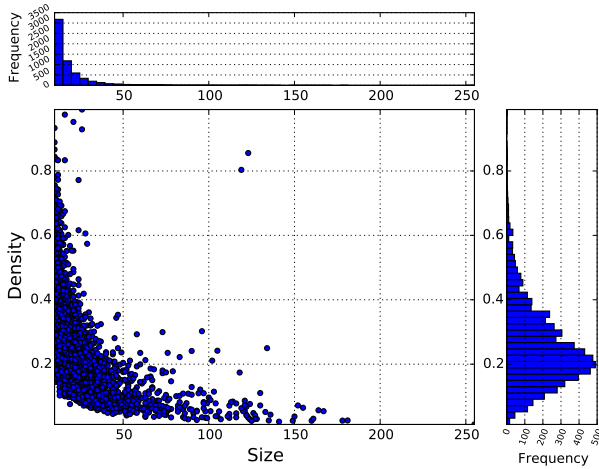


Figure 2.4: size, density distribution for networks for Github.com projects

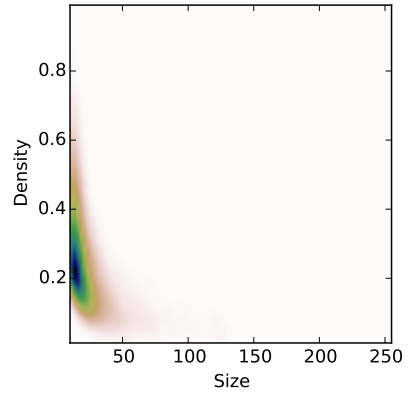


Figure 2.5: size, density probability kernel for Github.com projects

generate our random networks based on an estimated probability density function for the size and density. The probability density function is estimated by a kernel density estimator, visualized in Figure 2.5. We explained the limitations and our solution for simplification only for Github.com. For Apache Jira based networks the same limitations apply. Also the Apache Jira control group was sampled based on an estimated probability density kernel.

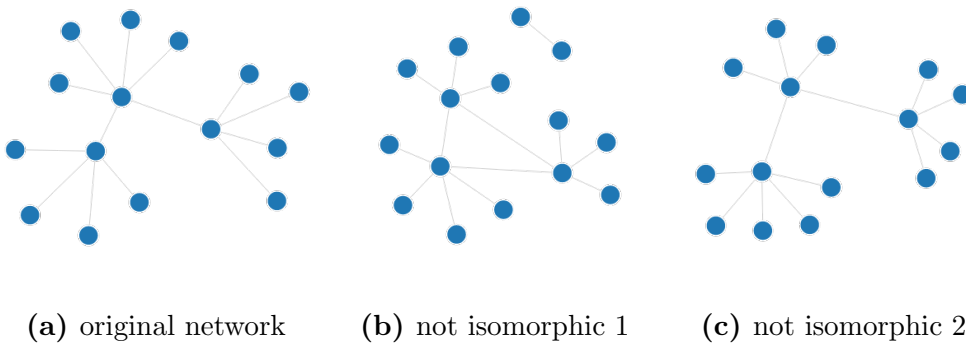


Figure 2.6: An example network (15 nodes, centralization=0.58) and its only two with link switching producible networks, which are not isomorphic.

2.4.4 Relationships

Direct property relationships

To examine direct relationships between different network properties and to get an understanding what might cause or influence the modularity and centralization results we analyze scatter plots and correlation coefficients for multiple network property pairs. For the correlation coefficients we use Spearman's rank correlation coefficient, since it does not require normal distributed variables and compared to the Bravais-Pearson-Correlation we can find monotonic relationships (Fahrmeir, Künstler, Pigeot & Tutz, 2007).

Temporal correlations

Since it is not just important whether any correlations exist, we also test on temporal correlations by comparing the binary course of two variables. It does not matter if it is a direct correlation, one variable increases and the other follows, or an indirect correlation, one increases and the other decreases. Furthermore a lag of one time frame backwards or forwards is considered. We take the variables for comparison, create their binary course and quantify the correlation by using the ϕ -coefficient (Fahrmeir et al., 2007) with time lags by one left, one right and no lag.

2.4.5 Efficiency

Kan, 2002 presented three categories for software quality metrics. One is called "maintenance quality metric" and is related to the maintenance phase after product release. The "Base on backlog management index (BMI)" measures how good support and developers handle bugs reported by customers. In OSS projects there usually is no clear separation of software development and a released product, but we can still use the BMI to measure the capability of handling issues as bugs are still reported and also additional issues such as feature requests are added to the issue tracking system. Capraro, 2013 adapted it as an efficiency metric and we do as well.

We slightly change the BMI to get negative values for time frames where fewer issues are closed than new ones are opened and a worst case value of minus one if no issues are closed at all. To get a symmetric metric, the maximum must be one, which corresponds to closing twice as many issues as new ones are opened. To gain this we rounded all values above one back to one as this occurs rarely and

the lost information is rather small. The open/close-ratio (oc-ratio) is formally defined as:

$$oc-ratio = \begin{cases} 0, & \text{if } (opened + closed) = 0 \\ 1, & \text{if } \frac{(closed - opened - reopened)}{(opened + closed)} > 1 \\ \frac{(closed - opened - reopened)}{(opened + closed)}, & \text{otherwise} \end{cases}$$

2.5 Used data sources

Capraro, 2013 developed a uniform integrated data model and presented how to normalize and store issue tracker data from Github.com, Jira and Bugzilla systems and developed a crawling system, which was capable of crawling issue tracking data from Github.com. To extend the data and to get a more representative set of data we extend his work by an additional crawler for Jira issue tracking systems, which is used by most of the Apache OSS projects.

Crawling from Github.com is problematic because of the limited amount of requests you are allowed to do, therefore you can not easily increase the amount of crawled projects. However the GHTorrent project presented a solution, as they are already crawling the Github.com data and made it available to the public (Gousios et al., 2014, Gousios, 2013). To keep the uniform crawling and processing system, we crawled the Github.com data by using GHTorrent data dumps until 2015-03-29.

We now have data for 16152150 Github.com projects and for 527 Apache Jira projects. Not all projects offer issue data, especially projects on Github.com, because the majority of projects are data dumps, forks, personal projects, etc., which are not used by other people. After applying the minimum edge weight filter (minimum edge weight of two), we only use networks with at least 10 people. The number of projects that can provide enough data to create such social networks are: 2037 projects providing 6007 networks for Github.com and 120 projects with 1198 networks for the Apache Jira issue tracking system. Based on these networks we do our social network analysis.

2.6 Research results

2.6.1 Centralization but no modularization

To make a statement about the general structure of OSS communities we study the probability density distributions for centralization and modularity for Github.com and the Apache Jira projects. The Figures 2.8 and 2.9 show that the centralization distributions are shifted strongly towards bigger values. The strong shifts are also confirmed by Mann-Whitney U tests and resulting p-values < 0.0001 , so that we can reject the Mann-Whitney U null hypothesis that both samples are from the same population. With the rejected Mann-Whitney U null hypothesis we can confirm our **hypothesis 1a** for Github.com and the Apache Jira projects having centralization values significantly different from random projects' centralizations.

	centralization			modularity		
percentile	5th	25th	50th / median	95th	75th	50th / median
Github.com	0.42	0.58	0.69	0.30	0.14	0.072
Apache Jira	0.33	0.54	0.66	0.48	0.24	0.15

Figure 2.7: percentile for the centralization and modularity distributions

For a more detailed assessment we look at the 5th and 25th percentile (Figure 2.7). 95 per cent of all networks have at least a medium centralization above 0.42 (Github.com) and 0.33 (Apache) and furthermore 75 per cent are strong centralized with values above 0.5. Thus, we can conclude that most OSS projects form a strong centralized community around a few key people.

Also for our modularity values the Mann-Whitney U test shows a significant shift for the Github.com and the Apache Jira projects with p-values < 0.0001 so that we can confirm **hypothesis 1b**, too.

In contrast to centralization the modularity distribution is smaller, meaning that the OSS projects are significantly trending to a lower modularity than random networks. Generally only networks with modularity values above 0.3 show a recognizable trend to form groups. Considered this and the 95th percentile, which has a score of 0.3, we can conclude that Github.com projects have a very low modularity (Figure 2.7). For the Apache Jira projects the result is not as strong as for Github.com but also there we have 75 per cent of networks with a modularity below 0.24. This findings conflict with related work and will be discussed later on. In summery most of Github.com as well as Apache Jira projects do not show a trend to distinct groups and both distributions are significantly lower than they are for random networks. Centralization, however, is much bigger for both issue tracking systems than for random networks, from which we can conclude that

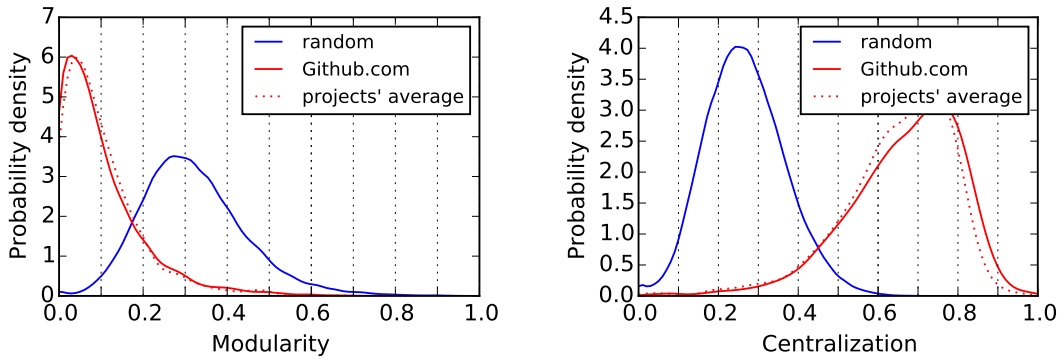


Figure 2.8: Github.com comparison of probability density functions

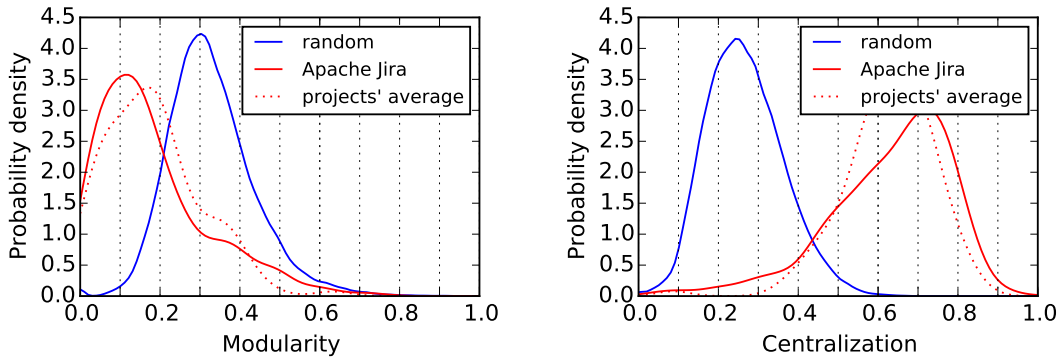


Figure 2.9: Apache Jira comparison of probability density functions

strong centralization is a typical characteristic for OSS communities. But this is not enough to accept hypothesis 2 and will be discussed in Section 2.6.2.

2.6.2 From centralization to modularization

In Table 2.13 it can be seen, that there is a significant correlation between modularity and centralization for Github.com and Apache Jira. If we take a look at the scatter plots and the regression line, the big cloud for the area with modularities below 0.3 makes the results somewhat unclear. Because of this we split the data based on modularity below 0.3 and equal or above 0.3 and repeat the correlation tests for both ranges.

The results in Table 2.10 show that the correlation is mainly contributed by networks with a modularity ≥ 0.3 and is even stronger than the correlation for all networks, although the correlation is a bit weaker for modularities < 0.3 . Also the scatter and regression plots, Figures 2.33 and 2.34, for the two

partitions do not provide any reason to reject the correlation. The regression lines point out a slight slope for modularities below 0.3 and a higher slope for a modularity ≥ 0.3 but in most cases with an increasing modularity causing a centralization decrease. We can conclude that it is unlikely for an issue tracking community to have a high modularity and high centralization at the same time, Thus, communities are either gathered around key members or they form mainly independent subcommunities without a strong leadership.

	Modularity		
	≥ 0	< 0.3	≥ 0.3
Github.com	-0.31	-0.26	-0.47
Apache	-0.35	-0.19	-0.46

Figure 2.10: correlations for modularity and centralization
all *p-values* < 0.0001

values of both. Network 2.35 is an example for a highly modular and also centralized network. It could be explained as a flat hierarchical structure with directly assigned workers to the head person. Quite the opposite is a fully connected network resulting in no centralization and a modularity close to zero. These two examples and outliers in our results show that the correlation is not dictated by the definitions and how they are calculated. Knowing this, we can accept **hypothesis 2**.

To determine if projects follow this correlation during their changes over time, meaning if a project increases its modularity the centralization drops, we use our approach for testing temporal correlations. The results in Figure 2.38 show no clear indications for a correlation over time in contrary to the general correlation of modularity and centralization. This phenomenon can be explained by looking at the differences, which occur between two time frames. The differences are rather small, mostly < 0.1 what can be seen in the Figures 2.11 and 2.12. With these small values an unpredictable behavior, e.g. modularity increases and centralization as well, could appear. At the same time they show that projects usually do not change their structure rapidly, But the most interesting fact is that apparently Apache Jira projects have considerably bigger differences between their maximum and minimum modularity and centralization values. The median of the maximum distance between the minimum and maximum values per project for modularity is 0.39 and for centralization 0.46. In contrast to Apache Jira, Github.com has a median maximum change inside projects of 0.16 for modularity and 0.28 for centralization. This also reflected in the Figures 2.14 and 2.15, where the Apache Jira projects show a bigger distribution for every age. This means most of the Apache Jira projects undergo stronger changes during

It could be argued that this correlation might have been foreseeable, because both measurements could be seen as mutually exclusive organizational types. Furthermore, both measurements are based on calculations using the same adjacency matrix. However a high centralization does not exclude a high modularity and vice versa and the same applies to low values

their total lifespan than Github.com projects do.

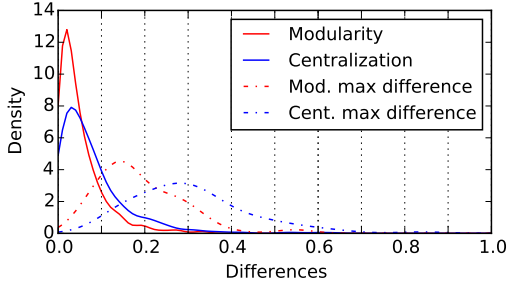


Figure 2.11: Github.com projects’ change differences from one time frame to another

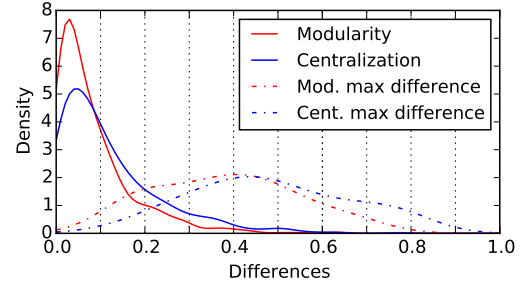


Figure 2.12: Apache Jira projects’ change differences from one time frame to another

2.6.3 Property relationships

The independence of network size

To find relationships we calculate Spearman’s rank correlation coefficient for the pairs listed in Table 2.13. One could conclude that the size might influence the structure, e.g. that bigger projects tend to have a stronger community modularization which turned out be a false assumption. The correlation coefficients for size paired with modularity and centralization actually show a low correlation (compare scatter and regression plots, Figures 2.17, 2.18, 2.19 and 2.20) in three out of four cases. The plots show a distribution where especially for low sizes the modularity is distributed on a large range. This creates a line leading to small correlation values but considering the whole plot we have to discard the correlation results. At most, what we can say is that high modularity values are generated by smaller networks. This means we have to reject **hypotheses 3a and 3b**.

In addition to a general correlation there is also the possibility of correlation over time, meaning that if the number of participants of a project changes the project structure changes, too. For verification we use the procedure to find temporal correlations presented in Section 2.4.4.

The results and probability density plots for temporal correlations for size and centralization are shown in Table 2.36. As we see the best results are achieved without a lag, but even those results do not show a clear correlation for size and centralization over time. In addition not enough projects show a significant p-value to accept a correlation.

We have already shown that modularity is not correlating with network size in general, and also the correlation density plots and p-values for temporal correlation regarding size and modularity also clearly show that there is no correlation over time (Table 2.37).

Consequently, we have to reject **hypotheses 4a and 4b**, as structure given by modularity and centralization is not changing in accordance to network size.

2.6.4 The development of projects

Slight changes in modularity and centralization

In the previous sections we showed that the organizational structure of projects of the two issue tracking systems vary over time. Now we want to analyze if this variation is connected to the age of a project, and whether they become more modular or more central over time. To do so, we take all projects with at least four quarters of activity and compare their modularity/centralization distributions at different stages in time (Figures 2.41, 2.42). Especially the distributions for Github.com show very stable distributions over time and the distributions for Apache Jira are only slightly less consistent. But both distributions are no indicators for older projects having a stronger/weaker modularity or centralization.

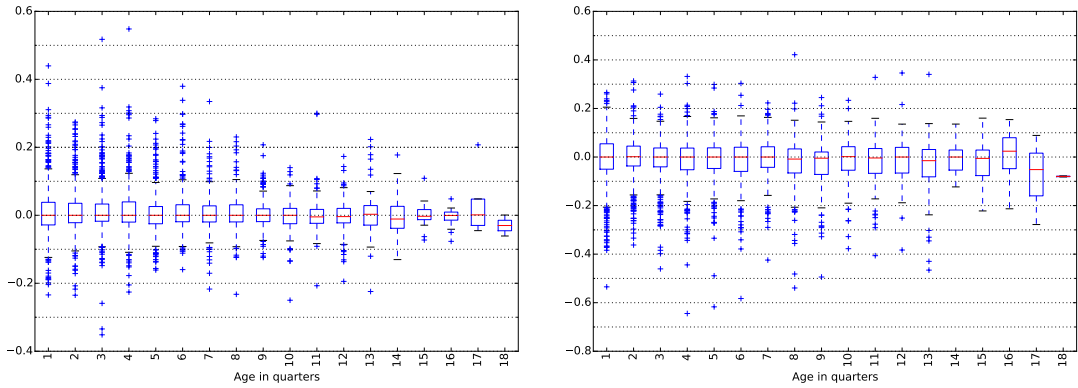
Also the Github.com projects' quarterly deviations from their medians (Figure 2.14) give no reason to think that there is a clear course of a project's structural development, rather do they show that projects deviate mostly symmetrical around their median modularity and centralization. For Apache Jira Projects (Figure 2.15) we can recognize that projects supervised by the Apache Foundation are slightly below their lifetime modularity median during their first quarters. Despite this slight increase for Apache Jira projects the answer to **ques-**

		modularity		centralization		open/close-ratio	
size	corr-coeff	0.17	0.07	0.19	0.26	0.02	-0.08
	p-value	0.0000	0.0224	0.0000	0.0000	0.0659	0.0070
communication amount	corr-coeff	-0.21	-0.33	-0.05	0.28	-0.13	-0.14
	p-value	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000
open/close ratio	corr-coeff	-0.04	0.13	0.19	-0.05		
	p-value	0.0053	0.0000	0.0000	0.1051		
centralization	corr-coeff	-0.31	-0.35				
	p-value	0.0000	0.0000				

columns:

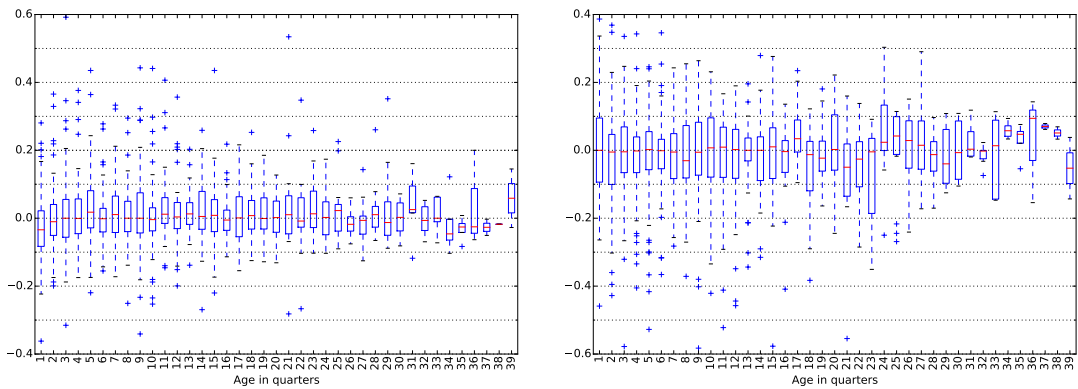
Github.com	Apache Jira
------------	-------------

Figure 2.13: network property correlations



(a) modularity median deviation distribution per past time frame (b) centralization median deviation distribution per past time frame

Figure 2.14: Github.com project's deviations from their projects' medians a given age (projects with at least four quarters)



(a) modularity deviations per age (b) Centralization deviations per age

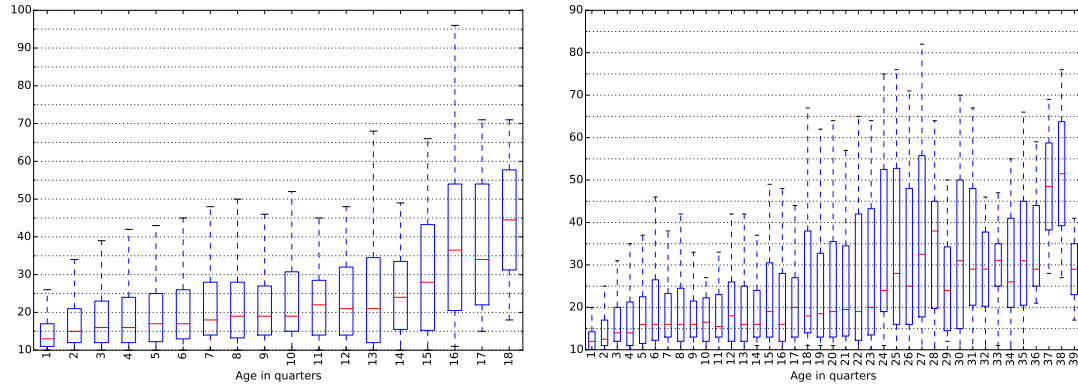
Figure 2.15: Apache Jira project's deviations from their projects' medians a given age (projects with at least four quarters)

tion 1 is that OSS collaboration networks do not increase their centralization or modularity over time. (Note: the results for very progressed projects get blurry because of the decreasing amount of projects.)

Growing projects

Besides the structural metrics we also look at size over time (Figure 2.16). We recognize from Github.com and Apache Jira projects that the median increases as well as the upper quartile does. The lower quartile is only rising slightly. This means that the median of projects get more active contributors over time

and therefore the size of networks increases. These results and the results from the previous section confirm again the independence of size from modularity and centralization, as both do not show the same behavior during the development of a project.



(a) Github.com size distributions depending on age (b) Apache Jira size distributions depending on age

Figure 2.16: Github.com and Apache Jira development of size for different project ages (projects with at least four quarters)

2.6.5 Structures and their efficiency

If efficiency could be related to network properties it would be helpful for OSS communities to improve their organizational structure. We use two methods to look for “good” properties. The first one includes general correlation tests for network properties and the open/close-ratio as it was done in previous sections. The results (Table:2.13) show small correlation coefficients for the oc-ratio and the total communication amount for Github.com and Apache Jira. But if we take the scatter plots (Figures 2.27 and 2.28) to validate these correlation values into account, they reveal the meaninglessness of these correlation values as well as the results for the open/close-ratio paired with modularity/centralization. Thus, we are not able to relate network properties to our efficiency values with this method.

For the second method we visually analyzed the kernel density plots for modularity - centralization with different filters applied, based on the open/close ratio. These filters are:

- the networks with the best open/close ratio for every project
- the networks with the top 100 open/close-ratios

-
- the networks with the worst open/close ratio for every project

The Figures 2.39 and 2.40 show the resulting plots. There might be minimalistic deviations but it is without any significance. Furthermore there are no significant changes if you look directly at the modularity versus centralization probability distributions.

Thus, both methods could not prove any relation between our efficiency metric and:

- modularity
- centralization
- communication amount

The answer to **research question 2** is that there are no organizational structures which showed a higher oc-ratio and therefore no organization type can be recommended to gain a higher efficiency.

2.7 Discussion

We wanted to explore the organizational structures and verify if other research results can be transferred from small or manually selected data sets to a larger, more representative one. We can confirm the findings made by Crowston and Howison, 2005 about the strong centralized structure of OSS communities. Thereby we discovered nearly no difference, whether looking at the many different kinds of projects on Github.com or looking at the more controlled ones by the Apache Foundation. Our results differ regarding the correlation of size and centralization. According to our results the size of a project does not have a correlation to any other network property, even though the average project increases its size over time. As it has already been addressed these centralization results could be beneficial for risk assessment as we can identify if key members leave a community. But it also shows that a few people have a lot of influence on projects, e.g. any discrepancies with one or a few people could make it impossible for developers or companies to cooperate on a project.

As the temporal development was not a central research point in related work, we were interested if changes of network properties are related to each other over time. Our results show that there are no temporal correlations between network size and any other properties like modularity and centralization. Based on these results and that most projects show a high centralization, we can conclude that OSS projects, regardless their size, show very similar organizational structures.

If we assume that collaboration behavior extracted from mailing lists is similar to extracting it from issue tracking systems, then our results are contradicting to the results found by Bird et al., 2008, because we found that OSS communities rather avoid the manifestation of distinct groups. It is more likely for random networks to form groups than it is for OSS projects, thus they even work contrary to the formation of subcommunities. This contradiction should be part of further research, including the influence of the divergence of our and their average network sizes.

As we were not only focused on one structural property, but we were calculating both, modularity and centralization, we investigated their relationships to each other. Our results show a strong correlation of modularity and centralization, but they do not show a temporal correlation. We trace the non-existing temporal correlation of both attributes back to the small interim changes and especially for Github.com to the small overall changes of the projects' structure. These small overall and interim changes show that OSS projects are lethargic in changing their organizational structure and due to the mostly symmetric deviations from a median they will likely cancel out the previously made changes later. This unexpected behaviour should be studied more deeply in future research. Also how strong the people inside the networks change while the community structure stays nearly the same.

A point, which also needs further research, is the influence of low active users. We filtered user connections based on edge weight and removed every edge with a weight below two. If we remove the edge filter, the resulting networks show a minor increased modularity and decreased centralization. The focus should be how important these users are and how exactly they influence communities.

2.8 Conclusion

We used issue tracking data from more than 2000 projects to create sequences of social networks for every project. We found that issue tracking communities for open source projects, central members play an important role and that they mostly avoid to form groups. Due to the high correlation of both we found, even if there is a high modularity it is unlikely to have a high centralization at the same time. Despite the correlation of modularity and centralization, networks do not follow this correlation during their temporal evolution. These findings are also owed to small transitions between different time frames and a small deviation of these network properties. Thus projects stay considerably stable although the size increases on average over time. Furthermore, we could not identify any patterns of modularity and centralization which show a significantly better efficiency in handling issue work load for the studied projects. This does

not make any organizational structure preferable to the other.

Appendix A Scatter plots for paired network properties

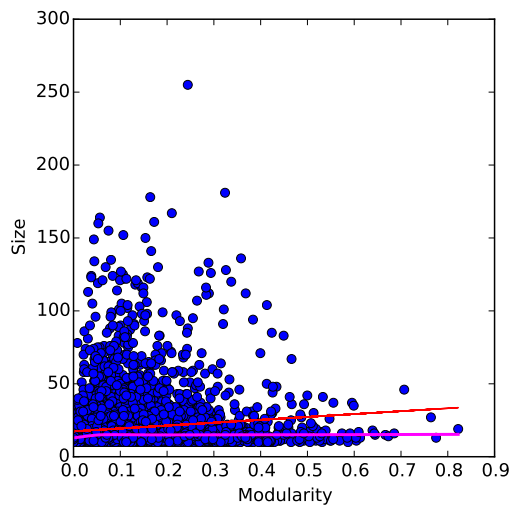
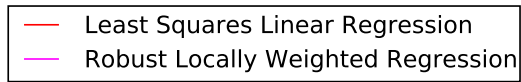


Figure 2.17: Github.com networks' modularity and size

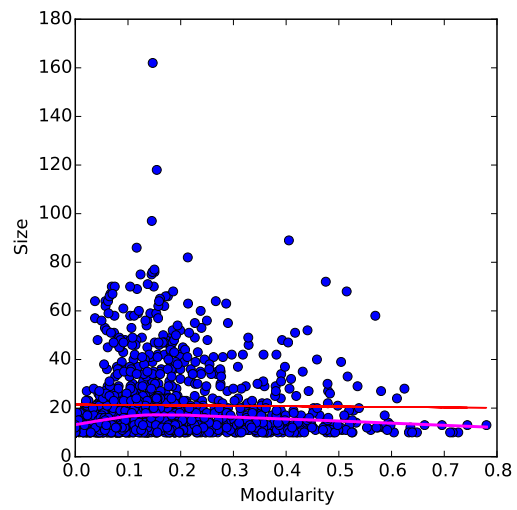


Figure 2.18: Apache Jira networks' modularity - size

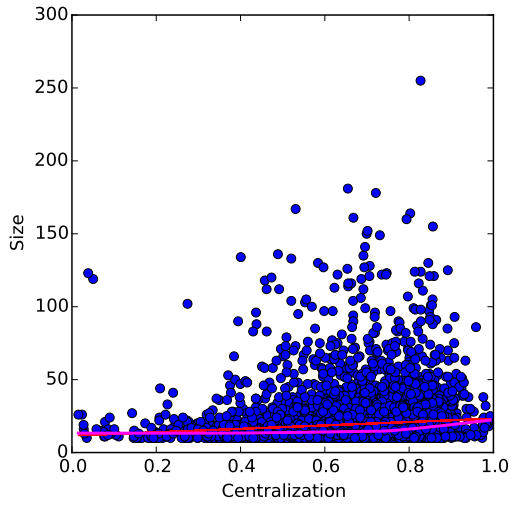


Figure 2.19: Github.com networks' centralization and size

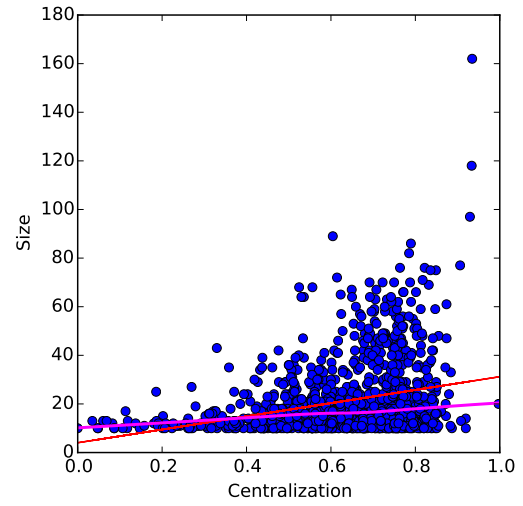


Figure 2.20: Apache Jira networks' centralization and size

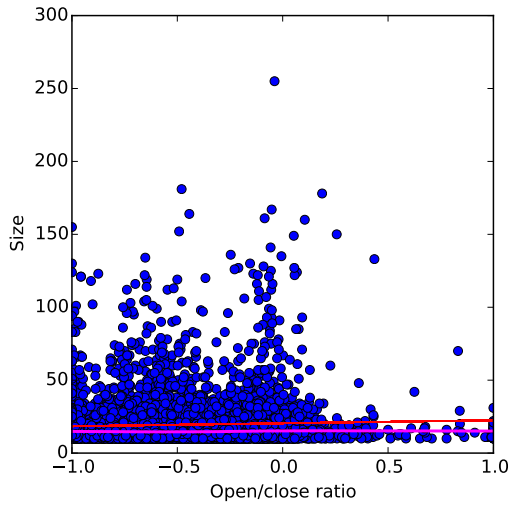


Figure 2.21: Github.com open/close ratio size

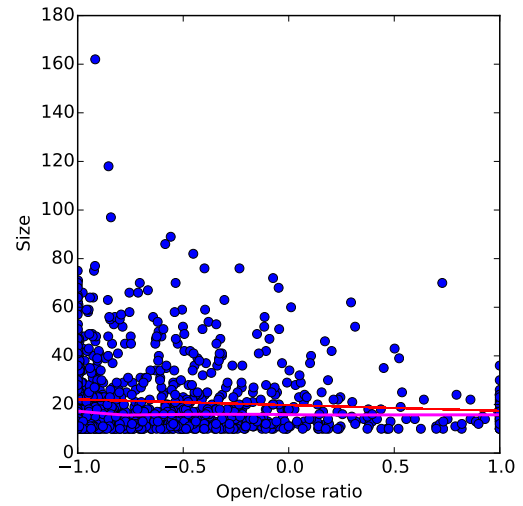


Figure 2.22: Apache Jira networks' open/close ratio and size

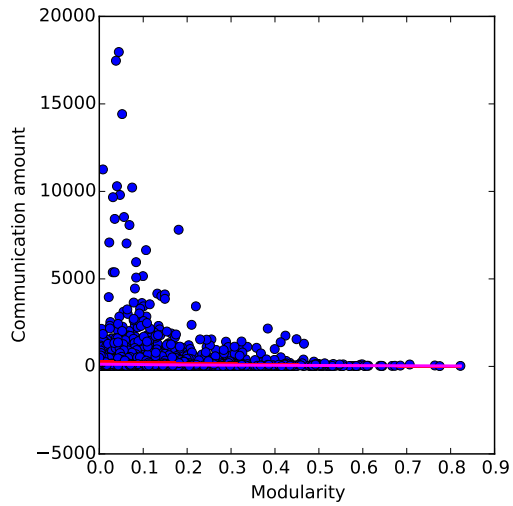


Figure 2.23: Github.com networks' modularity and communication amount

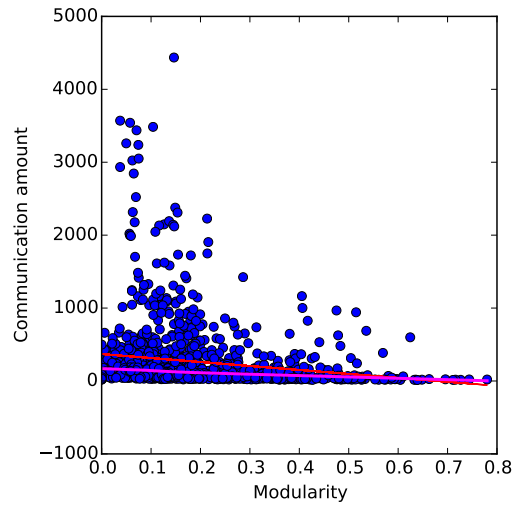


Figure 2.24: Apache Jira networks' modularity and communication amount

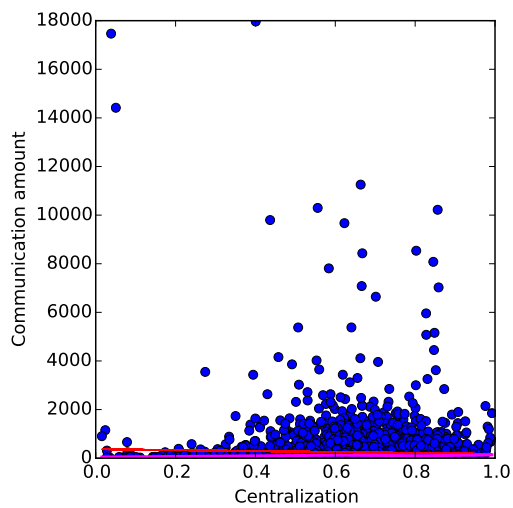


Figure 2.25: Github.com networks' centralization and communication amount

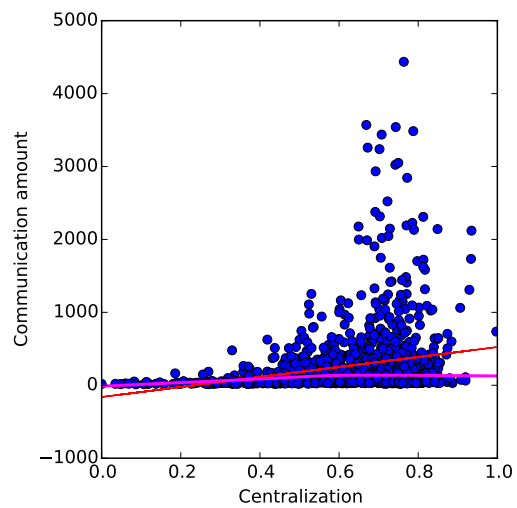


Figure 2.26: Apache Jira networks' centralization and communication amount

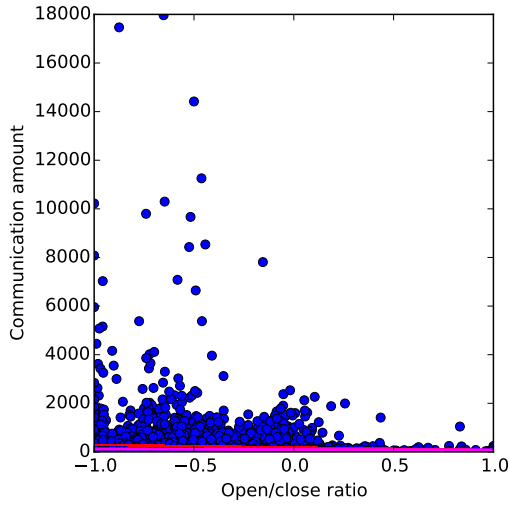


Figure 2.27: Github.com networks' open/close ratio and communication amount

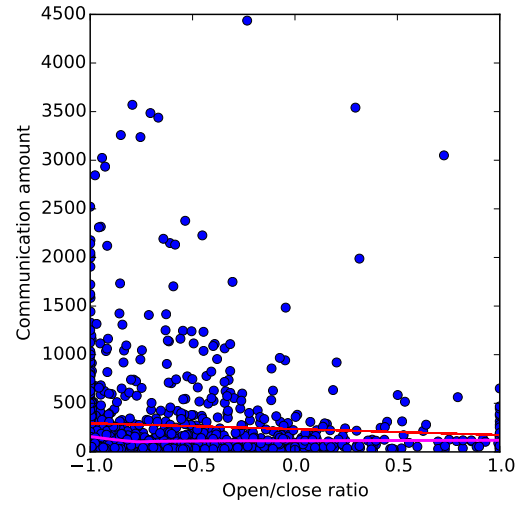


Figure 2.28: Apache Jira networks' open/close ratio and communication amount

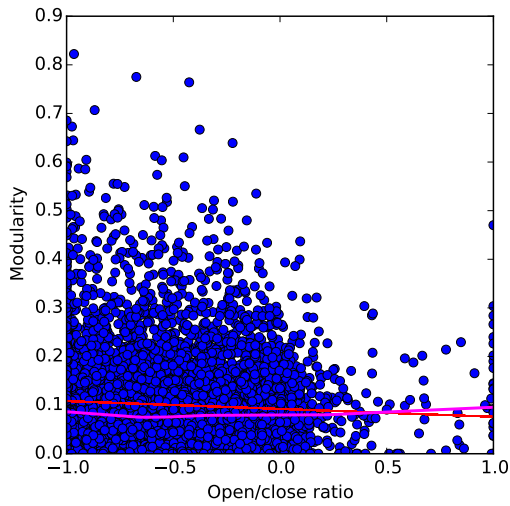


Figure 2.29: Github.com networks' open/close ratio and modularity

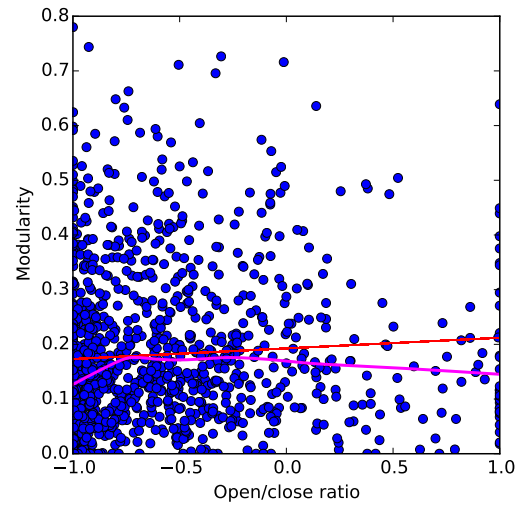


Figure 2.30: Apache Jira networks' open/close ratio and modularity

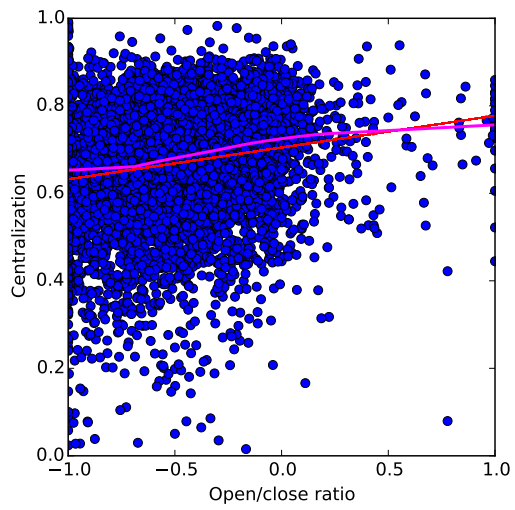


Figure 2.31: Github.com networks' open/close ratio and centralization

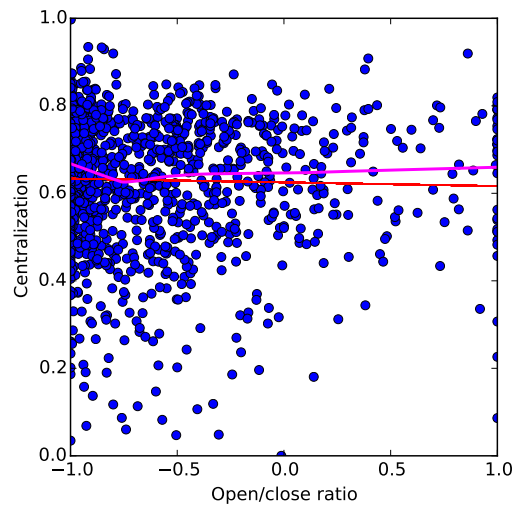
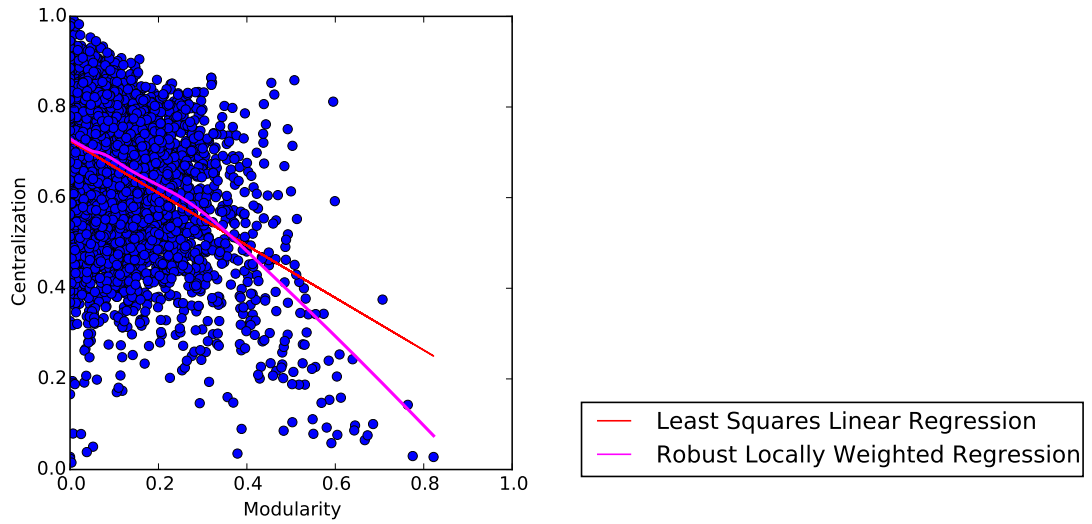
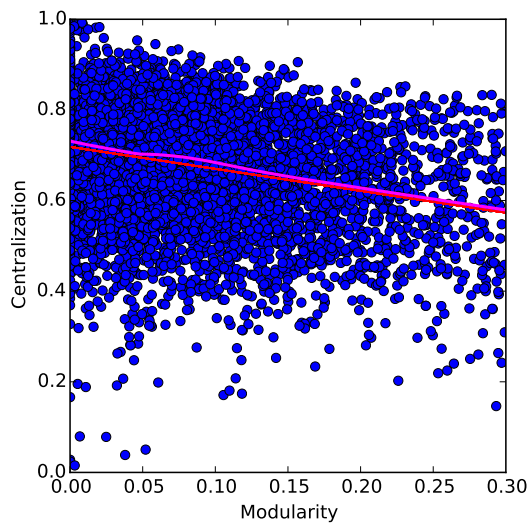


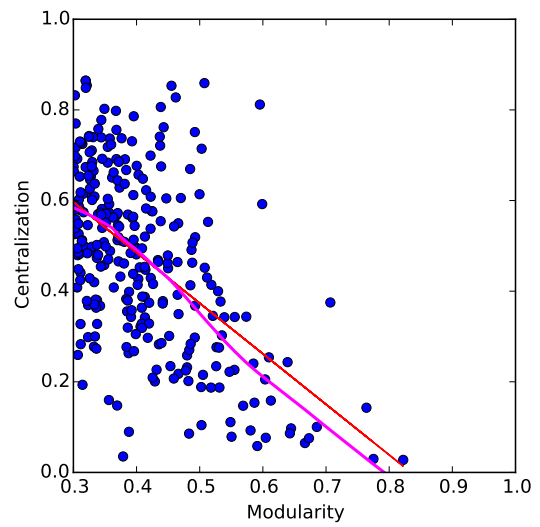
Figure 2.32: Apache Jira networks' open/close ratio and centralization



(a) modularity ≥ 0

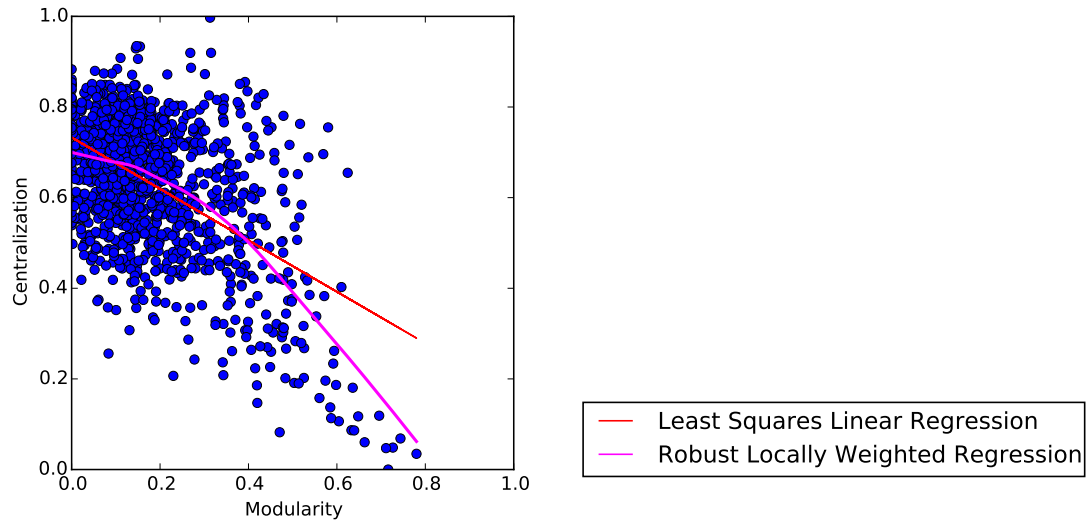


(b) modularity < 0.3

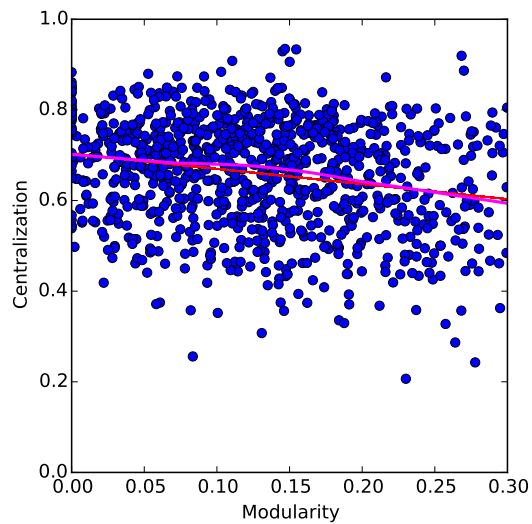


(c) modularity ≥ 0.3

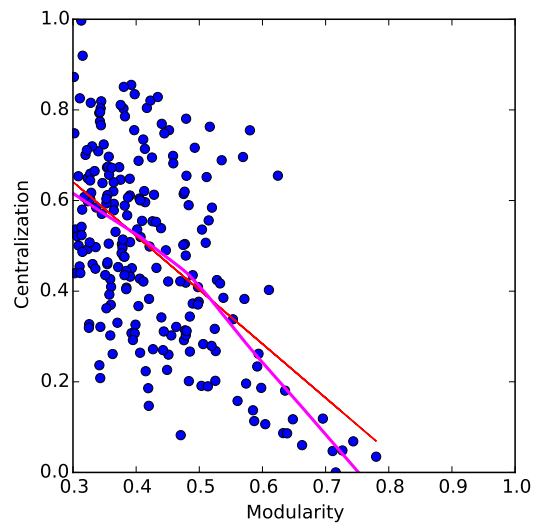
Figure 2.33: Github.com networks' modularity and centralization regression plot



(a) modularity ≥ 0



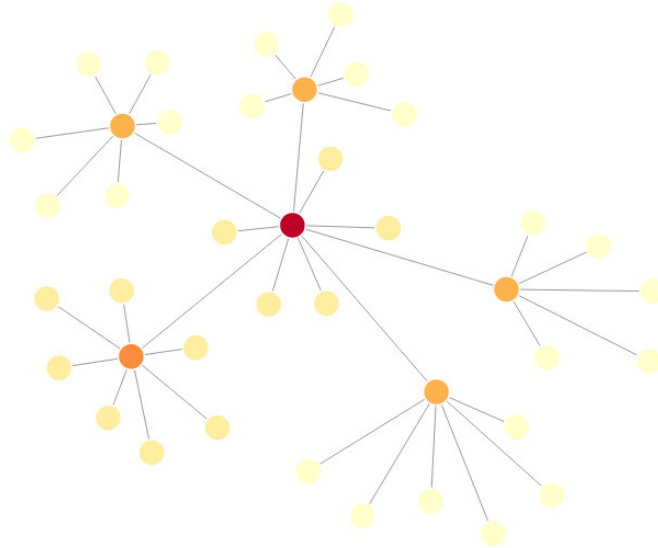
(b) modularity < 0.3



(c) modularity ≥ 0.3

Figure 2.34: Apache Jira networks' modularity and centralization regression plot

Appendix B Highly centralized and clustered networks



(a) central nodes, centralization: 0.65



(b) groups, modularity: 0.7

Figure 2.35: high modularity and high centralization network

Appendix C Temporal correlations for modularity, centralization and size

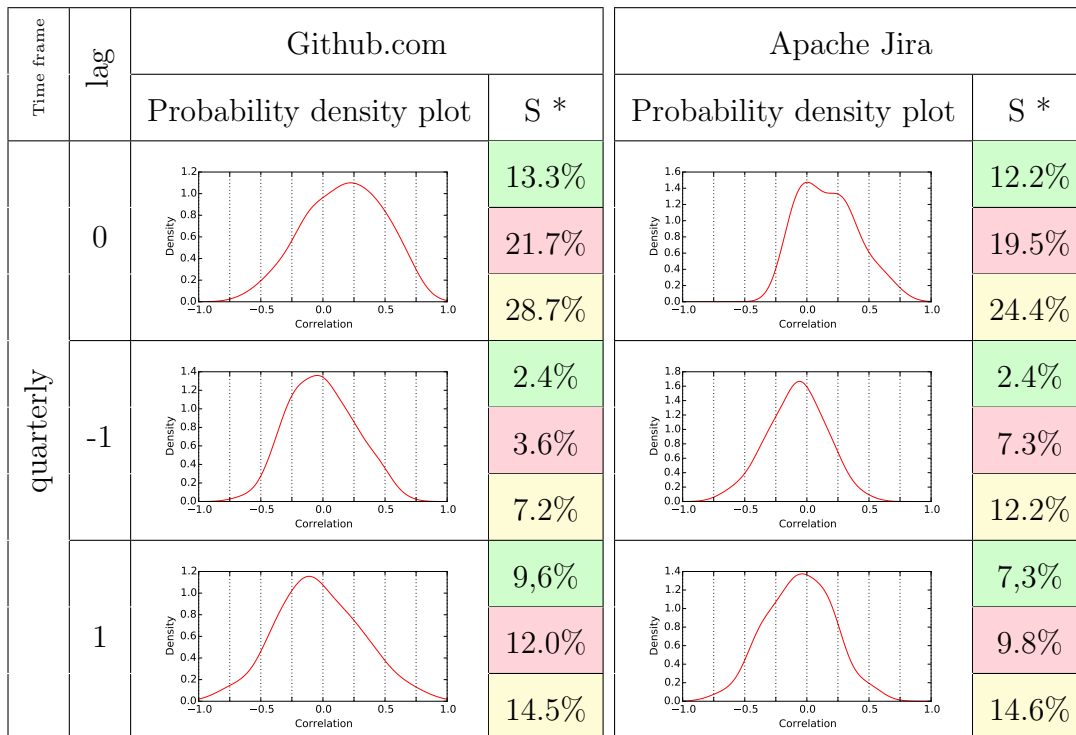


Figure 2.36: size and centralization time correlations for projects with at least 12 networks over time (Github.com: 83, Apache Jira: 41).

$$* S = \sum_{i=0}^n [p_i \geq sig] / n$$

[] Iverson brackets; p are the p-values; number of projects n

sig : significance levels: 0.05, 0.10 and 0.15

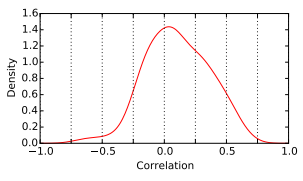
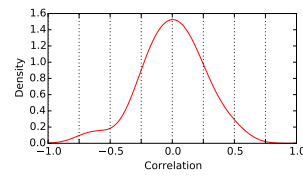
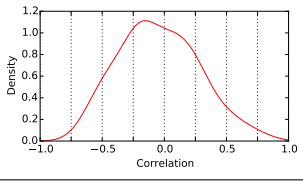
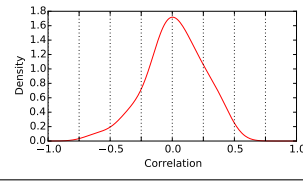
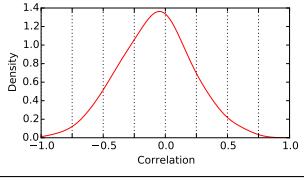
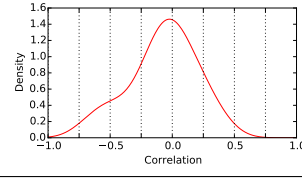
Time frame	lag	Github.com		Apache Jira	
		Probability density plot	S *	Probability density plot	S *
quarterly	0		3.6%		4.9%
			8.4%		7.3%
			15.7%		9.8%
	-1		4.8%		2.4%
			14.5%		9.8%
			18.1%		12.2%
1		3.6%		9.8%	
		9.6%		14.6%	
		12%		19.5%	

Figure 2.37: Size and modularity time correlations for projects with at least 12 networks over time (Github.com: 83, Apache Jira: 41).

$$* S = \sum_{i=0}^n [p_i \geq sig] / n$$

[] Iverson brackets; p are the p-values; number of projects n
 sig : significance levels: 0.05, 0.10 and 0.15

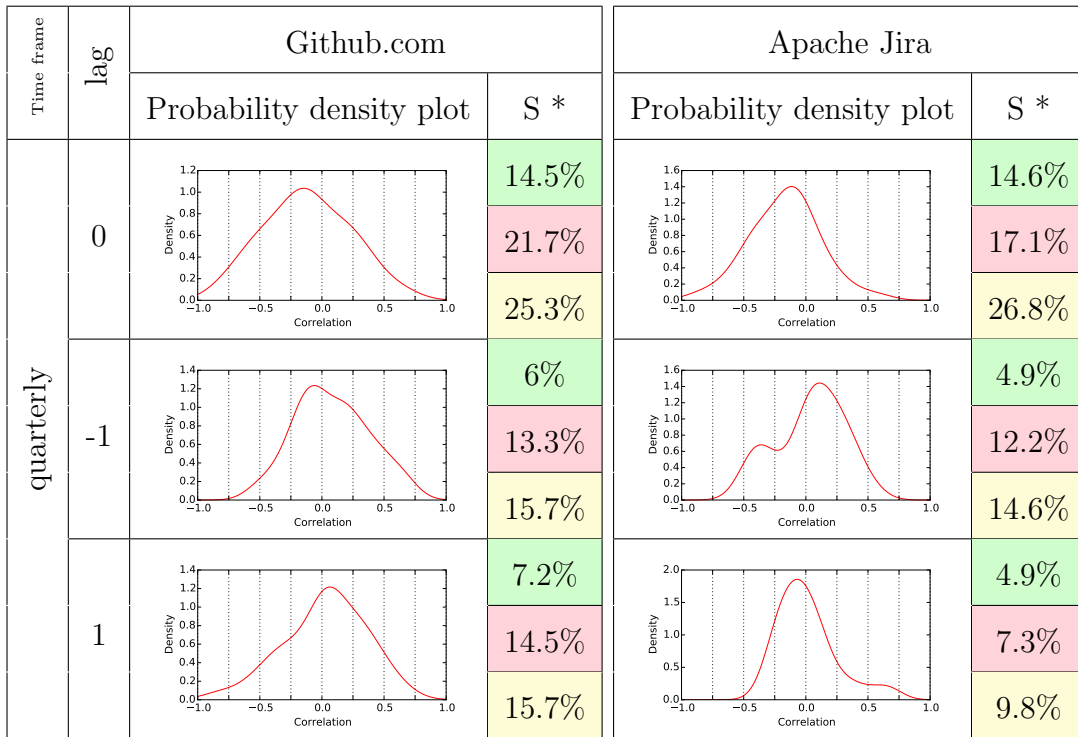


Figure 2.38: Modularity and centralization time correlations for projects with at least 12 networks over time (Github.com: 83, Apache Jira: 41).

$$* S = \sum_{i=0}^n [p_i \geq sig] / n$$

[] Iverson brackets; p are the p-values; number of projects n
 sig : significance levels: 0.05, 0.10 and 0.15

Appendix D Modularity and centralization density kernels based on efficiency

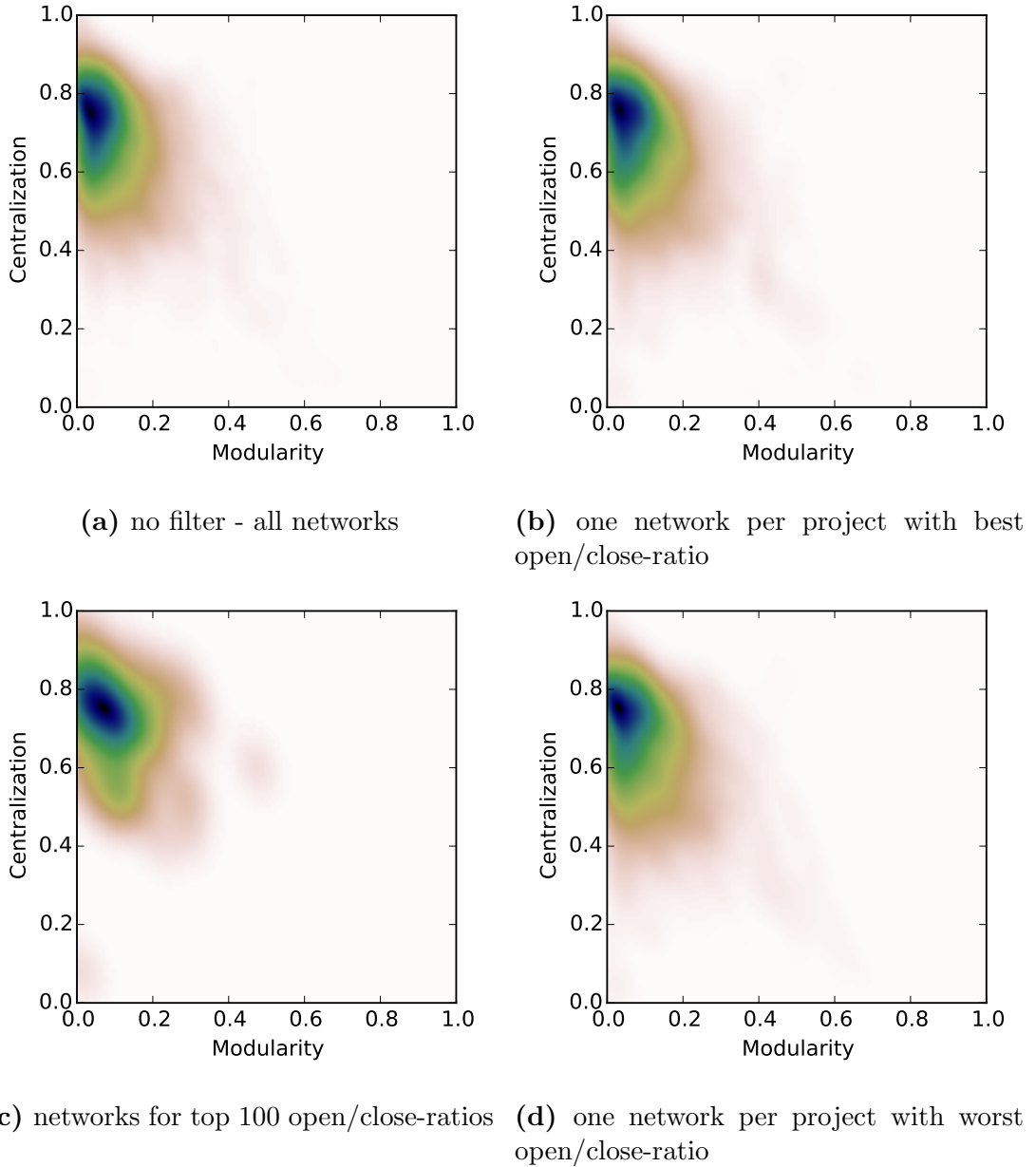
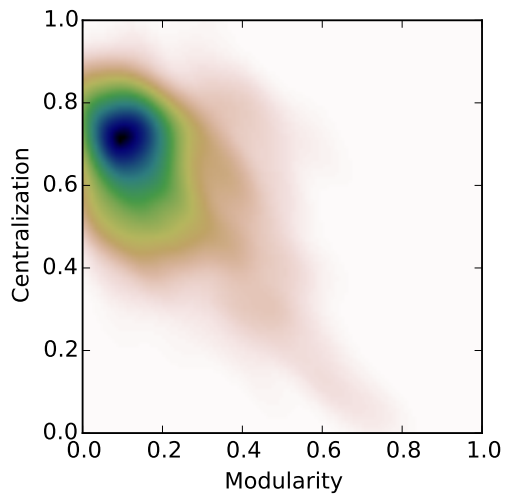
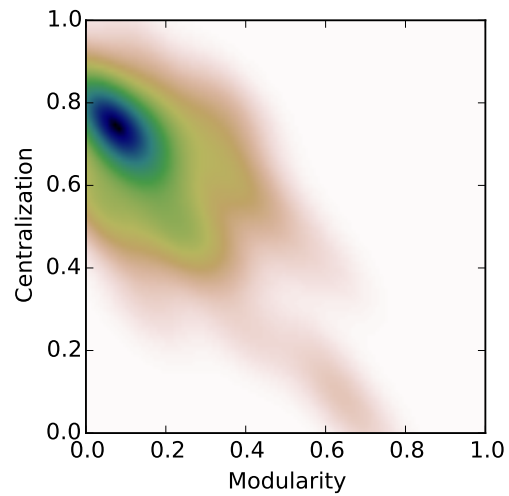


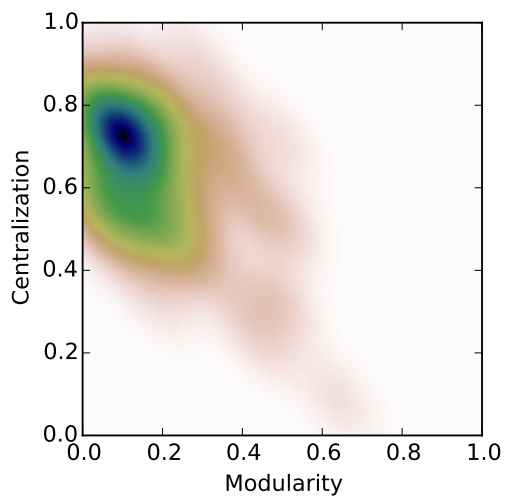
Figure 2.39: Github.com networks' modularity and centralization density kernels based on different open/close ratio filters



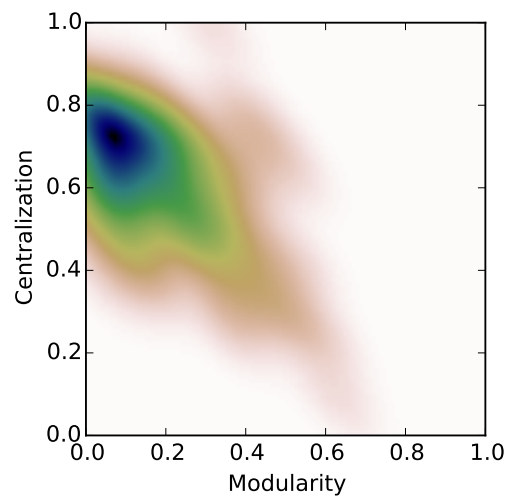
(a) no filter - all networks



(b) one network per project with best open/close-ratio



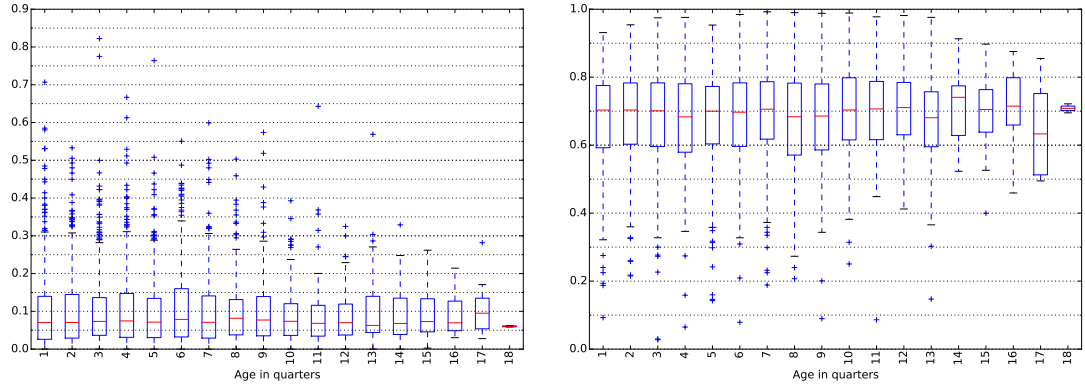
(c) networks for top 100 open/close-ratios



(d) one network per project with worst open/close-ratio

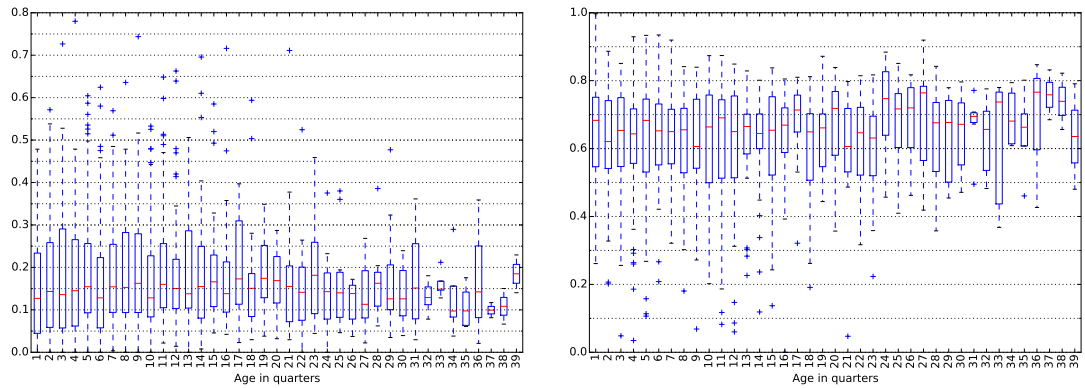
Figure 2.40: Apache Jira networks' modularity and centralization density kernels based on different open/close ratio filters

Appendix E Modularity and centralization distributions depending on project age



(a) modularity distribution per past time frames (b) centralization distribution per past time frames

Figure 2.41: Github.com modularity, centralization distributions for different project ages (projects with at least four quarters)



(a) modularity distribution per past time frames (b) centralization distribution per past time frames

Figure 2.42: Apache Jira modularity, centralization distributions for different project ages (projects with at least four quarters)

References

- Bird, C., Pattison, D., D'Souza, R., Filkov, V. & Devanbu, P. (2008). Latent social structure in open source projects. In *Proceedings of the 16th acm sigsoft international symposium on foundations of software engineering* (pp. 24–35). ACM.
- Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), P10008.
- Bollobás, B. (1998). *Random graphs*. Springer.
- Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z. & Wagner, D. (2006). Maximizing modularity is hard. *arXiv preprint physics/0608255*.
- Capraro, M. (2013). Towards a representative and diverse analysis of issue-tracker related code and process metrics.
- Crowston, K. & Howison, J. (2005). The social structure of free and open source software development. *First Monday*, 10(2).
- Crowston, K., Wei, K., Li, Q. & Howison, J. (2006). Core and periphery in free/libre and open source software team communications. In *System sciences, 2006. hicc's'06. proceedings of the 39th annual hawaii international conference on* (Vol. 6, 118a–118a). IEEE.
- Fahrmeir, L., Künstler, R., Pigeot, I. & Tutz, G. (2007). *Statistik: der weg zur datenanalyse*. Springer-Lehrbuch. Springer Berlin Heidelberg.
- Freeman, L. C. (1979). Centrality in social networks conceptual clarification. *Social networks*, 1(3), 215–239.
- Gousios, G. (2013). The gitorrent dataset and tool suite. In *Proceedings of the 10th working conference on mining software repositories* (pp. 233–236). IEEE Press.
- Gousios, G., Vasilescu, B., Serebrenik, A. & Zaidman, A. (2014). Lean gitorrent: github data on demand. In *Proceedings of the 11th working conference on mining software repositories* (pp. 384–387). ACM.
- Howison, J., Inoue, K. & Crowston, K. (2006). Social dynamics of free and open source team communications. In *Open source systems* (pp. 319–330). Springer.

-
- Kan, S. H. (2002). *Metrics and models in software quality engineering (2nd edition)*. Addison-Wesley Professional.
- Long, Y. & Siau, K. (2007). Social network structures in open source software development teams. *Journal of Database Management*, 18(2), 25.
- Molloy, M. & Reed, B. A. (1995). A critical point for random graphs with a given degree sequence. *Random structures and algorithms*, 6(2/3), 161–180.
- Newman, M. E. & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, 69(2), 026113.
- Sureka, A., Goyal, A. & Rastogi, A. (2011). Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis. In *Proceedings of the 4th india software engineering conference* (pp. 195–204). ACM.
- Thung, F., Bissyandé, T. F., Lo, D. & Jiang, L. (2013). Network structure of social coding in github. In *Software maintenance and reengineering (csmr), 2013 17th european conference on* (pp. 323–326). IEEE.
- Yu, Y., Yin, G., Wang, H. & Wang, T. (2014). Exploring the patterns of social behavior in github. In *Proceedings of the 1st international workshop on crowd-based software development methods and technologies* (pp. 31–36). ACM.