Revealing Governance Patterns by Taking Snapshots of Open-Source Communities

Alexandra Leta, Vrije Universiteit Amsterdam, Information Sciences

Abstract

The study of activity and interaction patterns between open-source community members enables software-related knowledge sharing and assists in efficiency improvement tactics. In this paper, we describe the extraction of organizational structure attributes values for metrics defined in [1], applied on public projects hosted on Github. Using these values and the defined thresholds we created 4 sets of repositories, corresponding to Formal Network, Informal Networks, Network of Practice and Informal Communities. We have defined 5 sets of quality metrics: group cohesiveness, social processes, workload allocation, project dynamics, social structure and determined their attribute values for the initial set of projects. We mapped the organizational structure attribute values and quality attribute values to understand better the relationship among software communities' practices and characteristics related to community types.

1. Introduction

Open source software communities are defined as a group of people who share similar goals, interests, beliefs, and value systems in a common domain of recurring activity or work **[2]**. This characterization is used by researchers to investigate groups of people working together and using information technology systems.

The study of open-source community characteristics and their mapping to different community types helps us understand the social structure of a community in order to be able to overcome different communication and collaborating barriers that may hinder a successful development process. A common method for acquiring knowledge on software communities patterns is to study the data provided by software tools used by community members. Previous studies on gathering data from software repositories have been conducted and their results were offered back to the community for further research purposes. The Flossmole project [3] provides a dataset which consists of source code metrics, meta-data including number of downloads or the employed programming language, structure and evolution metrics from several open source software projects.

2. Background & Motivation

Software projects continue to grow in size and complexity and an important requirement for studying snapshots of development communities is the acquisition and analysis of data from software repositories. Analysis of software communities and their evolution is needed to better understand the characteristics of software systems and projects and can be used to uncover knowledge and assist in software developments. Literature in software development communities already recognizes the need to study the effect of organizational structures on the quality of software, motivating further studies for understanding, representing and supporting social structures and communities **[3] [4] [5]**.

The relationship among software communities' practices and characteristics related to community types arises an interesting research problem as an emerging body of literature reveals open source software development as an alternative to proprietary software. The purpose of analyzing snapshots of open-source software communities is to reveal patterns in the software development process and social interactions that occur in these communities. This set of community characteristics and metrics could be used to understand how developers interact while working on a project and to study the effects of these social interactions under

conditions in which different projects have different levels of developer skill, and where features added by developers have the potential to influence other features.

A common problem identified in software repositories analysis is tracking developer identities across data sources as most software configuration management systems rely on user names to identify developers, while mailing lists and bug tracking software use the developers' emails. A possible solution to the problem is offered by Git, which uses emails to identify developers, while both bug descriptions and wiki entries coming from the same developers feature their identity details.

3. Research Questions

The number of people that manage to work together successfully to create high quality, widely used products is remarkably large. The research questions are aimed at understanding basic parameters of the process of creating open source software and how are these parameters related to the community type. The project aims to provide metrics on development practices in open source projects and to identify and understand patterns of development practices in this community. **What are the community and organizational patterns in open source development communities?**

In order to provide an answer for these questions the grounded theory research method is considered. Using this technique, the research findings constitute a theoretical framework "grounded" in the data, with built-in test of the theory incorporated into the results. The first step in analyzing data collected is to perform open coding, the process of breaking down, examining, comparing and categorizing the data **[6]**. The next step involves reexamining the data, looking for connections between categories.

4. Approach & Methodology

The main purpose of this research is to construct and analyze information and metrics on software development and their relation with software communities, using existing methodologies and tools. Software communities members use tools such as project's mailing lists, versioning systems, bug-tracking systems, discussion forums to enable peer collaboration in the software development process. This process leaves an accessible trail of data upon which interesting analyses can be built, but despite of the large set of data researchers often face significant challenges in using it. The Mining Software Repositories field analyzes the rich data available in software repositories to uncover actionable information about software systems and projects **[38]**.

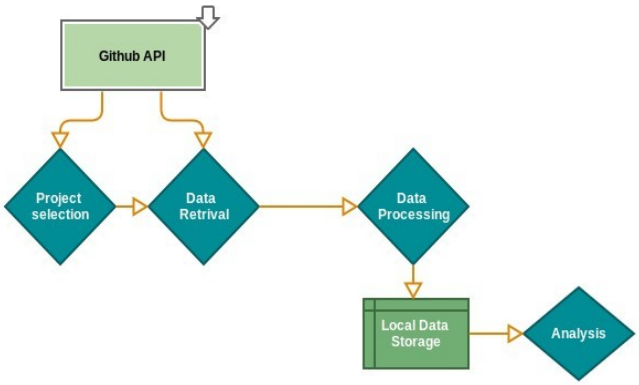

Figure 1. Constitutive steps of the research process.

This project is designed to gather collaborative data of open source software community, process and analyze this data in order to identify patterns that are most relevant for this community. We focus on community types that are most consistent with open-source development, namely: Formal Networks, Informal Networks, Informal Communities and Networks of Practice **[1]**. Using data gathered from software repositories and attached collaboration tools we analyze and compute a set of properties values for each analyzed repository. The same dataset and additional information extracted from the literature on social communities are used to define quality attributes and compute their accepted values for community types most compatible with open-source software in order to understand patterns in the open-source development process.

The results provided by this research process are reproducible, extensible and easy to compare when applied on multiple repositories. The open source software community development method poses a serious challenge to the commercial software businesses that dominate the software markets **[7]**

and this project creates the opportunity to exploit the diversity of existing research, to analyze the results in order to expand our knowledge regarding how community members interactions are related to the software development process.

4.1. Repository analysis

Members of the software development communities do not work in a single or central workplace, and often there is no formal management hierarchy in place to schedule, plan, and coordinate tasks and resources. Instead, developers contribute their effort to projects that they find significant, or otherwise professionally compelling. The snapshot data of this community includes data significant for the most common activities performed by its members: writing code and comments, fixing and submitting bugs, collaborating in order to find better solutions for their projects. GitHub Archive records the public GitHub timeline, archives it, and makes it easily accessible for further analysis and this archive dataset can be accessed via Google BigQuery.

In order to address the research question and understand how community members' interactions can influence the quality of the development process, we need key measures of project evolution and social interactions during the development process. The study of software developments snapshots includes creating basic summary reports about the state of the source code, as well as network analyses of open source development teams. Summary reports include the number of operations, chosen programming language, the number of developers per project, etc. This data is useful for creating general statistical information about open source projects.

GitHub provides access to its internal data stores through an REST API which researchers can use to access a rich collection of data and which enables retrieving commits to the projects' repositories and events generated through user actions on projects. Moreover, GitHub facilitates project contributions from non-team members, offers software forge facilities, such as a issue tracker, a wiki, and developer collaboration channel. The provided JSON format data is constantly updated and it allows running arbitrary queries and analysis over the entire dataset. This community data is used to compute attributes values related to the software development process, study some of the known phenomena observed in the open source software development communities, providing a better insight into their common practices and interactions.

4.2. Key attributes of the development community

We use metrics to increase our understanding of core software development activities and the overall nature of community participation in software projects. An important project quality metric is its complexity which can be used to predict the incidents of faults in code. A case study which used data derived from the change history of open source projects shows that change complexity metrics, based on the code change process, are reliable predictors of fault potential **[8]**. Similarly, researchers investigate how collaborative activities carried out by the developers such as social networks **[9]**, work dependencies **[10] [11]** and daily routines **[12]** are connected to the software product quality.

The following open source tools were used for the data extraction, processing and visualization:

- GitHub Java API: library for communicating with the GitHub API, supporting the GitHub v3 API. The client package contains classes that communicate with the GitHub API over HTTPS and the client is responsible for converting JSON responses to appropriate Java model classes. The package contains the classes that invoke API calls and return model classes representing resources that were created, read, updated, or deleted.
- Gephi Toolkit: Java library which provides useful and efficient network visualization and exploration techniques. The toolkit is a single JAR that could be used in applications and achieve tasks that can be done in Gephi automatically.
- Google Geocoding API: used for converting the addresses of repositories members into geographic coordinates, which is used to calculate distances. This API provides a direct way to access services via an HTTP request.

Based on the evidence from **[1, 5, 13]** the community types most consistent with open source software development communities are represented by: Formal Networks, Informal Networks, Informal Communities and Networks of Practice. Moreover, **[1]** provides a set of metrics defining the key attributes for these communities and a set of threshold values that are most representative for different community types using SourceForge project data.

The current approach uses Github repositories as data sources as Github has become the world's largest open source community. It provides an extensive REST API, which enables researchers to retrieve both the commits to the projects' repositories and events generated through user actions on project resources and has emerged as a popular project hosting, mirroring and collaboration platform **[14]**. For the selected Github repositories we computed the attributes values identifying the mentioned communities types, using metrics defined in [1]. The set of repositories is further analysed by defining a set of quality metrics and their values. Storing the processed data and running the analysis on updated collection of data will allow metrics to be compared over time, supporting historical comparisons and trying to make the analyses reproducible and extendable.

When conducting research with data from software repositories, an initial step involves retrieving of the project data to be analysed. The query defined in Table 1 was used to select the most popular repositories available on Github, considering the number of pull requests attached to the repository as a project reputation metric. GitHub Archive is a project which pushes its extensive collection of GitHub data to Google BigQuery. BigQuery in turn lets you write SQL-style queries on large datasets. The data generated from this activity can reveal interesting trends across many industries, including popularity of programming languages over time, defect rates, contribution metrics, and popularity of specific frameworks and libraries **[1]**.

4.2.1 Formal Networks

The most important attributes identified for **Formal Networks** are **formality** and **membership status**. Using the approach defined in **[1]** the attribute value formality can be computed using information regarding the number of milestones assigned to a project and the project lifetime.

```
SELECT repository_name,
repository_organization,  repository_url,
max(repository_forks) max_forks,
FROM [githubarchive:github.timeline]
WHERE repository_has_issues
    and repository_has_wiki
    and repository_watchers > 0
    and repository_organization!=""
GROUP BY repository_name,
repository_organization, repository_url
ORDER BY max_forks DESC
```

Table 1. Query used for repository selection. The vast majority of repositories hosted on Github are representative for open source software projects.

Only a limited number of projects hosted on Github have defined properties such as issues and milestones; our framework restricted the repository selection to projects that have these two properties. The membership attributes evaluates whether members and rigorously selected. For computing the value of this attribute we took into consideration the membership types assigned to a repository member: contributor and collaborator, as defined by the Github API. A contributor is member outside the core development team of the project who wants to contribute some changes to a project and a collaborator is a member on the core development team of the project and has commit access to the main repository of the project. The value associated to the hierarchy degree represents the percentage of contributors from the total number of repository members.
Using these attribute values and the thresholds defined in [1] we identified 9 repositories that matched formality and membership status thresholds from our initial set of project repositories. The set of repositories and their associated values are included in Appendix A.

4.2.2 Informal Networks

Informal networks are defined by properties such as **informality, openness** and **lack of governance.** We used an approach similar to **[1]** to compute the value of informality attribute. For each

repository member we can extract the repositories to which he has contributed, using the RepositoryService and company, in case we has filled this profile information. Using this collected data about users and their activity we computed metrics values for informality: average number of projects to which users have contributed and maximum percentage of users that are members of the same company.

The lack of governance attribute is computed based on the information regarding project milestones and project lifetime. As described in **[15]** open source communities approach to project milestones does not follow a common pattern. A comparison of release practices of important open source projects **[16]** analyzed properties such as dimensions of release authority, approval of releases and formats revealed considerable differences among projects. The results show that some projects have quite informal release schedules, following the pattern of releasing early and releasing often, whilst in other projects **[17]** releases come at an irregular rate.

The Github API imposes a restriction for requests regarding team components and team organization. Only users that are members of a specific repository can access this data, therefore the value for the **openness** attribute could not be computed.

Using these attribute values and their defined thresholds **[1]** we selected 5 repositories that matched informality, openness and non-governance thresholds from our initial set of project repositories. The obtained values are included in Appendix A. The average number of projects to which users have contributed metric was ignored because none of the considered repositories had an less than one, as defined by the thresholds.

4.2.3 Network of Practice

The set of defining attributes for **Network of Practice** communities includes the average distances, average cultural distance between community members, self-similarity and size.

Figure 2 presents the geographical distribution of members collaborating on one of the considered repositories. The red circles represent the location of community members collaborating on **[18]**. Edges representing connections between users are not included in this figure, but a connection graph is provided for step **4.3.5**. The same representation is available for each repository as most Github users have a location property defined.

Figure 2: Geographical distribution of users contributing to a repository

Using the Google Geocoding API we convert the addresses into geographic coordinates, which are used to calculate the **average distance** between community members. In addition to average distance attribute, cultural distance, self-similarity and number of active members are also key-attributes for **Networks of Practice**. We computed the values for **cultural distance** using the indices defined by Hofstede in a study on cultural dimension; for each community member who has the location property, we determined the index values for his country.

Github users profiles don't include their skills; based on the defined programming language for repositories to which a user has contributed we were able to determine a set of skills for each user and compute the **self-similarity** degree. Figure 3 represents the distribution of technical skills for community members of **[18]**. Python is the main programming language used for this repository, but a significant number of users have also contributed to repositories that use Ruby, Shell and C++ as their main programming language. The maximum percentage of users sharing a technical skill is used as a measure of self-similarity.

The CommitService allows retrieving the commits for a repository, limiting their number proportional to the maximum number of requests allowed by the Github API. Based on the list of commits we determined the number of users whose total number of commits contribution is equal to at least half of the commit activity. Using attributes identifying networks of practice we selected 13 repositories that matched the defined thresholds for their values. The number of active members metric was not taken into consideration because none of the considered

repositories didn't matched the defined thresholds.

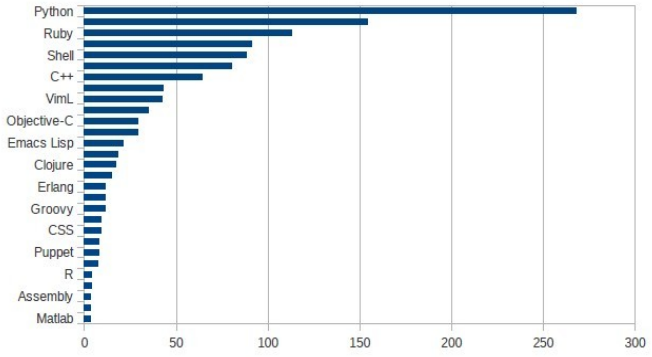

Figure 3. Community members technical skills

4.2.4 Informal Community

Based on the evidence from **[1], [13]** engagement, self-similarity and dispersion are key attributes defining an **Informal Community**. One indicator of high engagement within a community is the number of comments members post. Using the PullRequestService and CommitService resources we extracted comments posted by community members that were commenting of a pull request or adding a comment for a commit event. Using properties such a comment author and date we were able to compute the average number of comments posted by repository members. The values for self-similarity and dispersion were computed at step **4.2.3**. Attribute values and the set of repositories which matched the defined thresholds are included in Appendix A.

4.3. Qualities of the development community

After determining the values for key attributes of open-source community types, we selected a set of metrics to better assess the quality of different development communities. These metrics are group in 5 categories: group cohesiveness, social processes, workload allocation, project dynamics and social structures. Values and thresholds for these attributes were identified using the same set of Github repositories defined in the previous step and are available in Appendix A.

4.3.1 Group cohesiveness

We used the average number of user-follow events within community members and the average number of user connections established by collaboration on files as metrics of group cohesiveness.

Github is providing an easy accessible channel for software developers to share their projects and collaborate on them. This “social coding” environment allows developers to connect with each other based on common interests and projects. Activity traces of attention such as following, watching, and comment activity are an indicator that the community cares about that person, project, or action and lead to developer attention to that person, artifact or event **[19]**. When following another user, members will get notifications on their personal dashboards about their Github activity.

We considered followers associativity as an indicator of how closely community members interact, measuring preference to connect to other repository members. For each community member we determined the number of users that he follows or is followed by and are members of the same community, measuring preference to connect to other community members. According to **[19]** members follow the actions of other developers because they trust their coding skills, reporting interest in how they coded, what projects are working on and following. Frequent follow subscriptions indicate that group members are more inclined to participate readily and to stay with the group, trusting their collaborators and contributing to group cohesiveness.

An additional metric which quantifies how tightly community members collaborate on repository items is related to the number of community members that commit on common repository artifacts. For each repository contributor we determined the list of other repository members that have made changes on the same set or a subset of the files that the user committed to. Frequent collaborations on files between users indicates that they are willing to collaborate and possess linking bonds which increases social relations cohesiveness.

Goal	Purpose	Measure
	Issue	The group cohesiveness of
	Object	Repository members
	Viewpoint	From the external observer point of view
Question	Q1	Are users paying interest in their collaborators activity?
Metrics	M1	Average number of user-follow events within community members.
Question	Q2	Are users collaborating on repository items?
Metrics	M2	Average number of user connections established by collaboration on files.

Table 2. Group Cohesiveness

4.3.2 Social processes

We used the percentage of community members which publish blog posts, the average number of repository-watch events within community members and the average user activity time span as metrics for the social processes that occur in a community.

Social processes are dynamic interactions among community members as they work on a project **[15]**. To evaluate engagement of community members in social processes, we used data regarding their participation in activities such as blogging, repository watching and their committing longevity within the project. For each analyzed repository we computed the number of community members that have defined a **blog** property in the profile, the number of users who are **watching** the repository activity and the **time span** of members' activity.

Software development is not an isolated type of activity **[20]**. Known as an individual-intensive activity, coding is also complemented by other social processes and research work in **[21]** shows that a significant part of the developers working hours is spent interacting in various ways with other community members. Blogs are used as broadcast mediums, but also as a means of exchanging technical ideas or share knowledge on relevant topics. In **[21]** a blogger is defines as a "networker" who establishes social relations consisting of semantic references to allow for continuous communication, as well as of social references to express a social tie to another person.

Repository watching is recognized as a measure of community approval for a project and also it supports learning and acquiring technical knowledge. Visible information about community interest in the form of watcher count for a repository is an indicator that a project has a high quality. A Case Study on Social Coding in Github **[19]** indicates that community members use the number of watchers for a project as a signal that the project has community interest. We used the number of repository watch subscriptions as a metric for social processes that occur in the context of a development project.

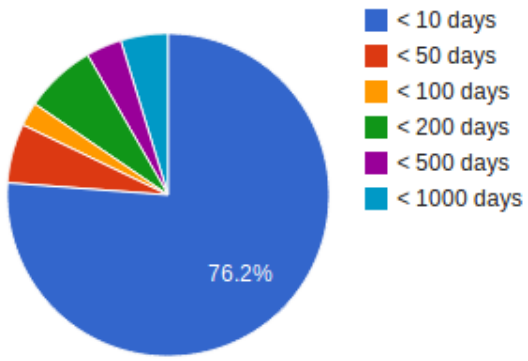

Figure 5: Commiter longevity

Committer longevity is an measure of how long one author remains part of the community. Moreover, **c**ommunity members contribution time span during the project lifetime indicates the knowledge level and his involvement within the community. The CommitService allows us to retrieve repository commits and for each commit we can extract information such as author, creation date, files that are affected by the commit. For each user we extracted the dates of his first and last commit within the community and determine the time span of his activity.

Figure 5 represents the relation between commiter longevity and the number of users who remain active for different periods of time; **[18]** is the repository used to compute the graph data. A large percentage of repositories members are active only

for periods shorter than 10 days and the project lifetime is shorter than 3 years.

4.3.3 Workload allocation

Number of commits pushed by each user, the set of users who collaborate on files through commits and the set of members whose contribution counts for a high percentage of the community workload are metrics that indicate how the project workload is divided among community members. These variables can be determined by retrieving and analyzing events that are generated by users' actions on project resources and they measure how responsibilities and collaboration are distributed across community members.

Goal	Purpose	Measure
	Issue	The team social characteristics
	Object	Repository members
	Viewpoint	From the external observer point of view
Question	Q1	Are users involved in interactions related to knowledge exchange?
Metrics	M1	Percentage of community members which publish blog posts.
Question	Q2	Do other users manifest their interest on the current repository?
Metrics	M2	Average number of repository-watch events within community members.
Question	Q3	How long authors remain part of the community?
Metrics	M3	Average user activity time span.

Table 3. Social processes

The case study presented in **[22]** shows that user involvement is a key concept in project development and has positive effects on project success and user satisfaction. The CommitService provided by the Github API allows retrieving the list of commits associated to each of the considered repositories. Each commit entity contains information about the person who initiated the commit event and we used attributes values to measure the number of commit events initiated by users. We used members commit contribution as a measure of individual productivity and community involvement.

We also examined the development activities of repository contributors to see if they were working together on common tasks. Collaboration on project files fosters interaction between community members and it is a valuable data source for tracking interactions. We investigated the repository file structure without storing the contents of files. ContentsService and DataService allow us to extract the repository file structure and obtain a list of commits history for each individual file. Based on this information we computed the number of community members which collaborate on repositories items and an median value of collaborators at the repository level. Figure 6 presents project **[18]** distribution of community members collaborating on repositories items. For the vast majority of repository items the number of community members which contribute to a file is less than 5 users.

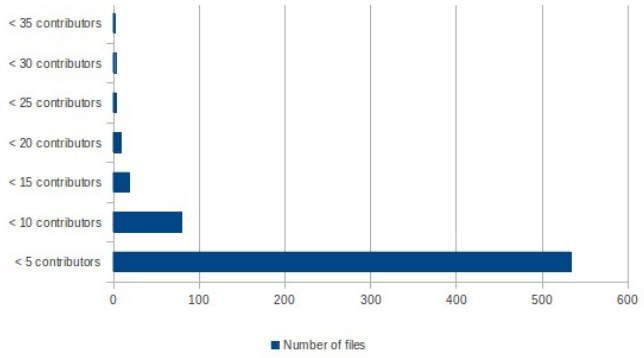

Figure 6: Number of individuals participating in the development of a repository item.

An additional metric which indicates how tasks are divided among repository contributors is the number of users whose contribution counts for at least 50% of the number of commits.

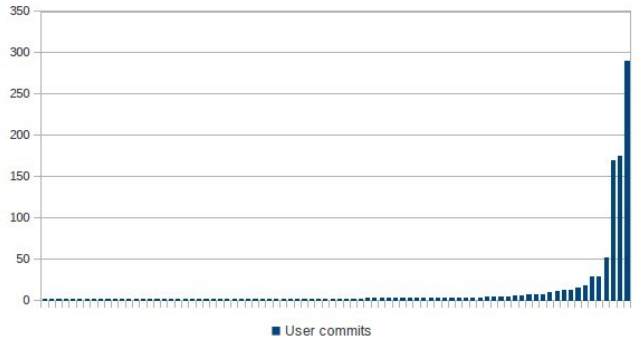

Figure 7: Community members commits contribution.

4.3.4 Project dynamics

We used the monthly evolution of issues, the percentage of re-opened issues from the total number of issues and the existence of publicly available Wiki as metrics for repository projects dynamics.

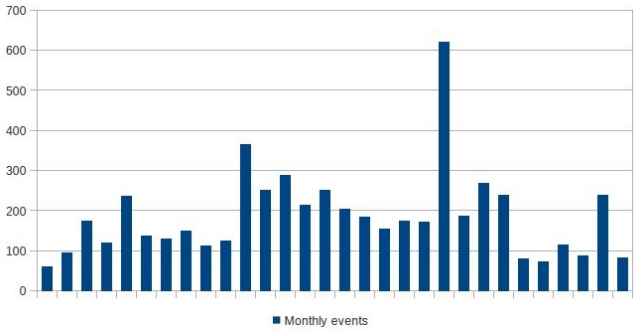

Figure 8: Monthly evolution of issue events

For the set of analyzed repositories we checked whether they have an attached Wiki. According to **[23]** project wikis can serve as a knowledge repository, a means for staging the project, a coordination mechanism, and a shared workspace. Repositories which contain a Wiki provide useful information on project artifacts and other resources that were included or linked in, ensuring ease to access to data. Moreover some repository wikis store historical information about the project, such as releases, project versions, providing a better insight into the dynamics of the project.

IssueService allows us to retrieve issue related events and we used it to compute the monthly evolution of these events. Events such as Open Issue, Close Issue, Merge Pull Request, Create Pull Request were divided using event creation date and we computed the standard deviation of the number of monthly events. Figure 8 represents the monthly evolution of issues related events for the **[18]** repository.

Issue reopen events after being closed are part of the software development process. [**24**] identifies some of the most common causes of issue reopen: developers initially did not understand the task purpose and as a result the issue was incorrectly closed, the issues did not have enough information to understand the bug and locate its root or the severity or impact of a bug has been underestimated. We focused on computing an estimation of reopened issues compared to the total number of issues.

4.3.5 Social Structure

We used the modularity, closeness centrality and clustering coefficient as measures of the community social structure.

The open source software development is a highly decentralized process and it is defined by volunteer effort where developers freely join projects that they find appealing. In order to analyze the social structure of open-source communities, we collected data regarding user collaborations using API requests to each analyzed repository. Based on the described data extraction mechanism, we obtained a network of nodes representing community members and edges representing a particular interaction between two members. The network graph representations are available in Appendix C and can be used for further analysis or they can be extended by adding new nodes and connections between them. We considered that two nodes are connected if at least one of the following conditions is fulfilled:

1. Common projects: two community members have at least one common repository to which they are contributing, except for the currently analyzed repository
2. List of followers: between the considered community members exists either a “is following” or “follows” relation
3. Pull request interaction: we consider the connection between the pull request author and other community members that are participating on the pull request

Goal	Purpose	Measure
	Issue	The project dynamics of
	Object	Project repository
	Viewpoint	From the external observer point of view
Question	Q1	Are project members ensuring ease of access to project information?
Metrics	M1	Existence of publicly available Wiki.
Question	Q2	How are issue events evolving throughout project life span?
Metrics	M2	Monthly evolution of issues.
Question	Q3	How often do issues get re-opened?
Metrics	M3	Percentage of re-opened issues from the total number of issues.

Table 4. Project Dynamics

Goal	Purpose	Measure
	Issue	The workload allocation for
	Object	Repository members
	Viewpoint	From the external observer point of view
Question	Q1	How is the workload divided among community members?
Metrics	M1	Number of commits pushed by each user.
Question	Q2	Are users collaborating on repository items?
Metrics	M2	Number of individuals participating in the development of one specific file.
Question	Q3	Are the repository actions divided equally among community members?
Metrics	M3	Members whose contribution counts for a high percentage of the community workload .

Table 5. Workload Allocation

Using the dataset of interactions we analyzed structural features for the user collaborations network. Using measures from social network analysis we can quantify how tightly community members collaborate and how resilient collaboration topologies are against the loss of central community members. Social network analysis has been applied for open source projects in **[25, 26]** but the present paper focuses on how the network measures are mapped to different open source community types.

A definition of "community" in the context of social networks is a subnetwork whose intra-community edges are denser than the inter-community edges. We used the algorithms provided by the Gephi library to compute the average clustering coefficient, modularity and closeness-centrality and visualize the network structure for the considered repositories.

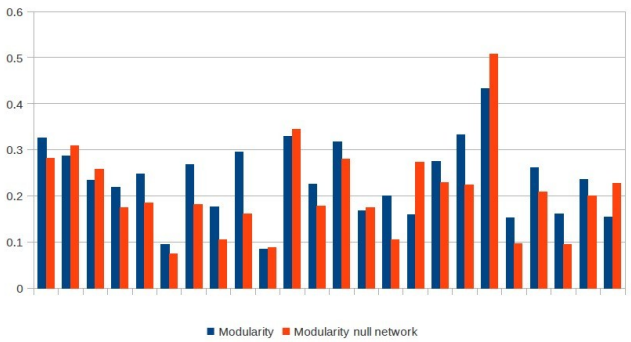

Figure 9. Modularity values for selected networks and their equivalent null networks

The community structure of a network is defined as the division of network nodes into subsets of nodes within which the connections are dense, but between which they are sparser. Modularity is an indicator of the strength of these sub-communities measuring how well a network can be divided into clearly defined subsets of nodes. Studies in **[27, 35]** show that community structure exists within the development communities of open-source projects because communities members spontaneously form sub-communities and communicate more with people within subgroups than outside them. The graphical representation in Figure 9 indicates that modularity values for the considered repositories and are higher than the modularity values obtained from random networks with the same number of edges and nodes, reflecting how community members associate. Clustering coefficient is a statistical property which measures how complete

the neighborhood of a node is **[28]**. A key element in defining the clustering coefficient for a node is a triangle which, in the context of an undirected graph, is a set of three vertices such that each possible edge between them is present in the graph. The clustering coefficient value is computed by dividing the number of triangles containing the vertex by the number of possible edges between its neighbors. The **clustering coefficient** of the whole graph represents the average of this value for all vertices and in order this value to be meaningful it should be significantly higher than in version of the network where all of the edges have been randomized.

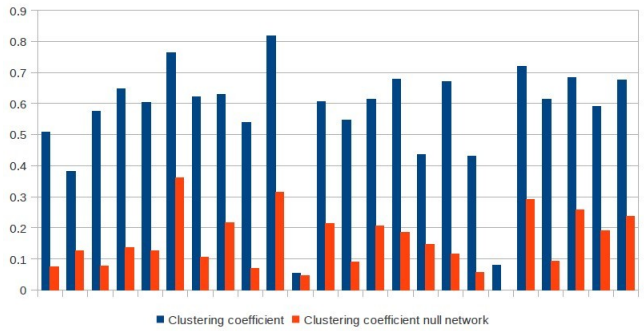

Figure 10. Clustering coefficient values for selected networks and their equivalent null networks

A common pattern of social networks is the tendency for connections to be transitive, or for friends of friends to be also connected **[29, 31]**. The clustering coefficient of a network quantifies this notion and studies indicate that social networks exhibit larger clustering coefficients than would be expected from their overall edge density alone **[29][30]**. In social networks, the tendency for new connections to be made through mutual connections are cited as forces towards high clustering coefficients **[31]**. The graphical representation of clustering coefficients in Figure 10 shows that clustering coefficients for the considered repositories and are higher than the clustering coefficients for their equivalent null networks. Community members tend to create tightly groups characterized by a relatively higher density of ties than the average probability of a tie randomly established between two nodes.

Closeness centrality is defined as the average distance from a given node to all other nodes in the network and measures how close each community member is to all other users **[32]**. Centrality indices are based on shortest paths linking pairs of users, measuring the average distance from other users and the ratio of shortest paths an user lies on. High centrality scores indicate that a vertex can reach others on relatively short paths, or that a vertex lies on considerable fractions of shortest paths connecting others **[33, 34]**.

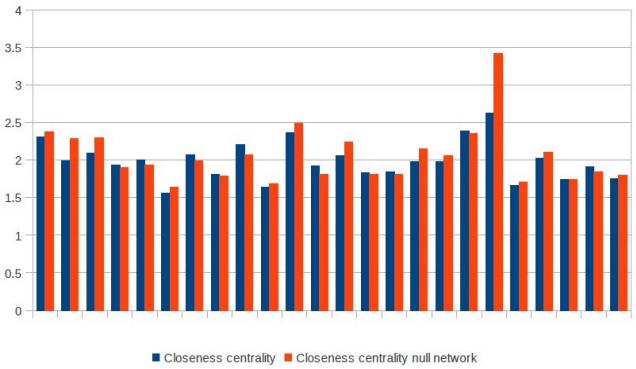

Figure 11. Closeness centrality values for selected networks and their equivalent null networks

Figure 11 shows that community members can reach others on short paths, closeness centrality values ranking between 2 and 3. Ease of access between graph nodes is also determined by the fact that active users are connected to a large number of community members. We computed the **average degree** nodes for each of the considered repositories as the average number of edges that are adjacent to the community node.

To understand better how social graphs representing community members interactions differ from graphs randomly generated we created a network which matches the original network in some of its topological features, but which does not display community structure. Figures 9, 10 and 11 display the values of social structure metrics of the initial network and the corresponding values in the null model. The null model preserves the number of nodes and connections between them, but the connections are randomized, obtaining a **randomized** version of the original network.

Goal	Purpose	Measure
	Issue	The social structure of
	Object	Project members
	Viewpoint	From the external observer point of view
Question	Q1	Are the community members divided in subgroups with dense connections within them and sparse connections between them?
Metrics	M1	Network modularity.
Question	Q2	Which is the degree to which nodes in a graph tend to cluster together?
Metrics	M2	Clustering coefficient.
Question	Q3	How closely do nodes in the graph communicate with other nodes ?
Metrics	M3	Closeness-centrality.

Table 6. Social Structure

6. Discussion

The first step in analyzing open source software development communities was computing the values for key-attributes described in chapter 4.2, attributes which are most representative for open-source communities. Taking into consideration the thresholds of these attributes as defined in [1] we identified:

- 9 repositories that matched the Formal Network thresholds for formality and membership status
- 5 repositories that matched the Informal Network thresholds for informality, openness and non-governance
- 13 repositories that matched the Network of Practice thresholds for dispersion, self-organisation, self-similarity, openness and size
- 9 repositories that matched Informal Community thresholds for self-organisation, self-similarity, openness and dispersion

For the Informal Community key-attribute, engagement, according to [1] each member of an informal community should post no less than 30 comments per-month. As presented in Figure 5 a large percentage of users have an activity time span value smaller than 10 days, as many repository contributors have a low degree of engagement, contributing only for a short period. We ignored the engagement attribute when selecting the range of repositories which matched the identified thresholds for a high level of engagement.

For the 4 different subsets of repositories, each one corresponding to a different community type, we applied the set of metrics defined in chapter **4.3** which describe group cohesiveness, workload allocation, project dynamics, community social structure and social processes that occur within these communities. The obtained values are included in Table 7.

For the set of metrics describing **group cohesiveness**, the average user-follow events that occur within community members has similar values for all considered community types. However, formal networks display a slightly lower number of user-follow events, indicating that community members prefer less to connect to other repository members within communities with a high hierarchy degree. The second metric which describes community cohesiveness indicates that members of formal networks establish more connections through collaboration on files than informal network members do.

By analyzing the **social processes** which occur within different community types, we observed that members of an informal network express more interest in publishing blogs. This behavior can be explained by taking into consideration that a blogger is defined as a “networker” (a person who establishes social relations consisting of semantic references to allow for continuous communication) and that the key-attribute of an informal network is represented by the informality of communication between members.

	Formal Network	Informal Network	Network of Practice	Informal Community
Group cohesiveness				
Average user-follow events within members	0.008	0.019	0.012	0.012
Average user connections established by collaboration on files	0.224	0.143	0.182	0.184
Social processes				
Percentage of members which publish blog posts	0.568	0.729	0.638	0.656
Average repository-watch events within members	49.669	30.555	35.565	31.682
Average user activity time span	86.812	77.041	100.965	95.010
Workload allocation				
Number of commits pushed by each user	28.506	13.528	23.514	28.271
Number of individuals participating in the development of one specific file	2.349	3.789	2.505	2.372
Member whose contribution counts for a high % of the community workload	2.111	1.8	2.25	1.875
Project dynamics				
Existence of publicly available Wiki	True	True	True	True
Monthly evolution of issues	286.13	250.26	259.58	265.49
Percentage of re-opened issues from the total number of issues	0.47	0.96	1.21	0.92
Social Structure				
Network modularity	0.255	0.139	0.243	0.226
Clustering coefficient	0.525	0.717	0.602	0.62
Closeness-centrality	2.058	1.705	1.988	1.921

Table 7. Attribute values for Formal Network, Informal Network, Network of Practice and Informal Community

In the set of metrics regarding **workload allocation**, the values for average number of commits pushed by each user are similar for the considered community types. Nevertheless, this metric is not representative for any of the considered community types. The next metrics – number of members whose contribution counts for a high percentage of the community workload – indicates that a small group of repository members is responsible for a large range of commits. Although the selected projects where rated with a high popularity based on the number of forks, the number of users whose contribution counted for at least half of the commits was smaller than 5 for all considered repositories.

By studying metrics of **projects dynamics** we observed that only 0.08% of the analyzed repositories didn't have a Wiki property defined. Taking into consideration that the analyzed repositories where selected among the most popular public repositories hosted on Github, we can state that is a common pattern for repositories to provide useful information on project artifacts and other resources that were included or linked in, ensuring ease to access to data. The percentage of re-opened issues from the total number of issues indicates less re-opened issues for formal networks.

Social structure metrics have similar values for all considered community types. Figure 9 indicates that **modularity** values for the considered repositories are significantly higher compared to modularity values of random networks of associations, indicating a good level of collaboration. The closeness centrality values indicate that there are better connections between community members compared to a community with random associations or communities. Clustering coefficient is one of the metrics that would require further analysis, as the differences between the clustering coefficient values of community networks and their associated null

models are not significant.

One of the advantages of having access to Github data resources through REST API calls is that it allows fast data retrieval for all repositories that are shared publicly, without being forced to entirely check out this data. However, among the disadvantages we can consider the limit of 5000 requests per hour for authenticated users and the restricted access to data regarding team members and status for users that are not registered as team members.

6. Conclusion

In this paper we have presented an approach to study metrics that capture organizational- structure and quality attributes of communities that are representative for open-source software development.

The project provides support for collecting communication and development data on open-source software projects hosted on Github. The process of collecting data is applied on a range of 25 distinct software repositories. For each repository we compute the values of 12 key-attributes that are relevant for formal networks, informal networks, networks of practice and informal communities. Based in the attribute values and the thresholds defined in [1] we group repositories based on their characteristics. For the initial set of repositories we compute a set of metrics which describe social processes, social structure, workload allocation, project dynamics group cohesiveness. The next contribution is determining a mapping between the quality metrics and the communities to which they correspond, getting an overview of the value ranges for some metrics suits.

The results indicate that only a relatively small number of the considered quality metrics have significant differences among the 4 communities types associated with open-source projects. The study of governance patterns by taking snapshots of open-source communities provides valuable data for a software analyst allowing him to: characterize the project in terms of social structure and and the activities performed by community members and compare different open source community types in terms of quality metrics.

The set of key-attributes that are frequently associated with open-source communities and the attributes measuring the project quality can be further extended and applied on a larger larger number of projects for a better understanding of the relationship among software communities' practices and characteristics related to community types.

Appendix A: Attributes values

The list of analyzed repositories and organizational structure and quality attributes can be found here: **[URL1]**. The data set includes attribute values for software repositories, the name and version of the project, meta-data and additional graphs.

Appendix B: Source code

The source code used for analyzing the repositories and attributes in Appendix A can be found here: **[URL2]**.

Appendix C: Graphs

Network graph representations for the considered repositories can be found here **[URL3]**.

7. References

[1] D. A. Tamburri, E. Di Nitto, S. Gatti, R. Galoppini, P. Lago, H. van Vliet, Supporting Community and Awareness Aspects in Open-Source Forges, Transactions on software engineering, Vol. XX, April 2013

[2] M. S. Elliott, W. Scacchi, Free Software Developers as an Occupational Community: Resolving Conflicts and Fostering Collaboration, in the GROUP '03 Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work, Pages 21 - 30, ACM New York, NY

[3] J. Howison, M. Conklin, K. Crowston, FLOSSmole: A collaborative repository for FLOSS research data and analyses, International Journal of Information Technology and Web Engineering 1(3), 17 (2006).

[4] H. Barkmann, R. Lincke, W. Lowe, Quantitative Evaluation of Software Quality Metrics in Open-Source Projects, Proceeding WAINA '09 Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops, Pages 1067-1072 , IEEE, 2009

[5] D. A. Tamburri, P. Lago, H. Van Vliet, Organizational social structures for software engineering, ACM Computing Surveys, 1-34, 2012

[6] S.P. Wainwright, Analysing data using grounded theory, Nurse Researcher, 1(3), 43-49, 1994

[7] A. Mockus, R. Fielding, J. Herbsleb, A Case Study of Open Source Software Development: The Apache Server, in Proceedings of the 22nd International Conference on Software Engineering. New York: ACM Press, 2000.

[8] A. E. Hassan. Predicting faults using the complexity of

code changes. In Proceedings of the 31st International Conference on Software Engineering, ICSE '09, pages 78–88, Washington, DC, USA, May 2009. IEEE Computer Society.
[9] T. Wolf, A. Schroter, D. Damian, T. Nguyen, Predicting build failures using social network analysis on developer communication, in ICSE '09: Proceedings of the 31st International Conference on Software Engineering. IEEE Computer Society, 2009, pp. 1–11.
[10] M. Cataldo, A. Mockus, J. A. Roberts, J. D. Herbsleb, Software dependencies, work dependencies, and their impact on failures, IEEE Transactions on Software Engineering, vol. 35, no. 6, pp. 864–878, 2009.
[11] N. Bettenburg, A. E. Hassan. Studying the Impact of Social Structures on Software Quality, ICPC, page 124-133. IEEE Computer Society, (2010)
[12] J. Sliwerski, T. Zimmermann, A. Zeller, When do changes induce fixes?, in Proceedings of the 2005, International Workshop on Mining Software Repositories. ACM, 2005, pp. 1–5.
[13] D.A. Tamburri, P. Lago, H. van Vliet. Uncovering latent social communities in software development. Software, IEEE, 30(1):29–36, Jan.-Feb.
[14] G. Gousios, D. Spinellis, GHTorrent: GitHub's data from a firehose, in MSR '12: Proceedings of the 9th Working Conference on Mining Software Repositories, June 2-–3, 2012. Zurich, Switzerland.
[15] K. Crowston, K. Wei, J. Howison, A. Wiggins, Free/Libre Open Source Software Development: What We Know and What We Do Not Know, Journal ACM Computing Surveys (CSUR) Surveys Homepage archive, Volume 44 Issue 2, Article No. 7, ACM, 2012
[16] J. R. Erenkrantz, Release Management Within Open Source Projects, Proceedings of the ICSE 3rd Workshop on Open Source Software Engineering (May 3, 2003).
[17] D. G. Glance, "Release Criteria for The Linux Kernel", First Monday 9(4), 2004
[18] Support of full breadth and depth of Amazon Web Services, https://github.com/boto/boto, Accessed August 2013
[19] L. Dabbish, C. Stuart, J. Tsay, J. Herbsleb, Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository, in Proceedings of CSCW '12 Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, Pages 1277-1286 , ACM , 2012
[20] D. E. Perry, N. A. Staudenmayer, L. G. Votta, Understanding Software Development, Processes, Organizations, and Technologies, IEEE Software - Software , 1994
[21] J. Schmidt, Blogging Practices: An Analytical Framework, Journal of Computer-Mediated Communication, Volume 12, Issue 4, pages 1409–1427, July 2007
[22] S. Kujala, M. Kauppinen, L. Lehtola, T. Kojo, The role of user involvement in requirements quality and project success, Requirements Engineering, IEEE International Conference on, 0:75–84, 2005
[23] B. R. Krogstie, The wiki as an integrative tool in project work, Proc. 8th Intl. Conf. on the Design of Cooperative Systems COOP08, page 111-122, 2008
[24] T. Zimmermann, N. Nagappan, P. J. Guo, B. Murphy, Characterizing and Predicting Which Bugs Get Reopened, Proceeding ICSE 2012 Proceedings of the 2012 International Conference on Software Engineering, Pages 1074-1083, IEEE Press Piscataway, 2012
[25] J. Howison, K. Inoue, K. Crowston. Social dynamics of free and open source team communications, Open Source Systems, pages 319–330, 2006.
[26] R. Nia, C. Bird, P. Devanbu, V. Filkov, Validity of network analyses in open source projects, In proceedings of the 7th IEEE Working
[27] C. Bird, D. Pattison, R. D'Souza, V. Filkov, P. Devanbu, Latent social structure in open source projects, In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, SIGSOFT '08/FSE-16, pages 24–35, ACM, 2008
[28] M. Latapy, Main-memory Triangle Computations for Very Large (Sparse (Power-Law)) Graphs, Journal Theoretical Computer Science archive, Volume 407 Issue 1-3, Pages 458-473, 2008
[29] M. O. Jackson, Social and Economic Networks, Princeton University Press, 2008.
[30] M. Brautbar, M. Kearns, A Clustering Coefficient Network Formation Game, Proceeding SAGT'11 Proceedings of the 4th international conference on Algorithmic game theory, Pages 224-235, Springer-Verlag Berlin, 2011
[31] D. Easley, J. Kleinberg, Networks Crowds and Markets: Reasoning about a Highly Connected World, Cambridge University Press, 2010
[32] A. Bavelas, A mathematical model for group structure. Human Organizations, 7:16-30, 1948
[33] U. Brandes, A faster algorithm for betweenness centrality, The Journal of Mathematical Sociology, 25(2):163–177, June 2001
[34] U. Brandes. A faster algorithm for betweenness centrality, Journal of Mathematical Sociology, volume 25, pages 163–177, 2001.
[35] B. Furht, Handbook of Social Network Technologies and Applications. Springer, 1st edition. edition, November 2010
[36] V. D. Blondel, J. L. Guillaume, R. Lambiotte, E. Lefebvre - Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment, Vol. 2008, No. 10
[37] M. E. J. Newman. Modularity and community structure in networks., Proceedings of the National Academy of Sciences, 103(23):8577–8582, June 2006.
[38] A. E. Hassan, The Road Ahead for Mining Software Repositories, in Proceedings of Frontiers of Software Maintenance, 2008. FoSM 2008.
[39] B. Doll, I. Grigorik, Analyzing Millions of GitHub Commits, what makes developers happy, angry, and everything in between