

The Open Source Software Community Structure

Jin Xu
University of Notre Dame
jxu1@nd.edu

Scott Christley
University of Notre Dame
schristl@nd.edu

Gregory Madey
University of Notre Dame
gmadey@nd.edu

Abstract

Social network properties are applied in many research areas to help people discover the intrinsic mechanism of a social organization. The social network analysis has recently been used to study the OSS development community [Xu et al. 2004, Xu1 et al. 2003, Xu2 et al. 2003, Madey et al. 2004, Lopez et al. 2004]. In this paper, we identify the community structure for the SourceForge project network. Furthermore, we examine software categories to explore possible reasons for the formation of the community structure. Our research provides useful information to study the interaction between projects and the communication and information flow in OSS virtual community.

Contact:

Jin Xu
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN46556

Tel: 1-574-631-7596
Fax: 1-574-631-9260
Email: jxu1@nd.edu

Key Words: Open Source Software, social network

Acknowledgement: Thanks to SourceForge for providing us data
Support: This research was funded in part by the NSF Award-0222829, from the Digital Society & Technologies Program, CISE/IIS.

The Open Source Software Community Structure

Jin Xu, Scott Christley, Gregory Madey

One network property that has been attracted increased interest is *community structure*. In some networks, some nodes are grouped together by a high density of edges, while there are few edges between those groups. These tightly-connected groups are called *communities* in a network [Newman2 et al. 2004]. Detecting communities in networks can reflect some important characteristics of networks. This method has been applied in many networks to find related vertices. For example, communities in a scientific collaboration network reflect scientists grouped together by similar research topic or methodology [Girvan et al. 2002]; communities in a gene network identify functionally related genes [Wilkinson et al. 2004]; communities in an email network represent actual social relationships [Tyler et al. 2003].

Open Source Software (OSS) development community can be modeled as a social network. Detecting communities in the OSS network have profound effects: first, we can find projects which might have related subjects, similar programming environment, or common developers. By identifying these communities, we can study their interaction during their growth; second, we can get information about the communication path and knowledge flow within or between communities. Such information can help us adjust and improve the robustness of communications in OSS.

This paper is an extension of our previous social network analysis [Xu et al. 2004, Xu1 et al. 2003, Xu2 et al. 2003, Madey et al. 2004] on SourceForge [SourceForge]. In this paper, we explore the community structure existed in the OSS project network. This network is constructed based on data collected and extracted from a SourceForge 2003 data dump. We apply community structure identification algorithm on the largest component of the project network. Moreover, we give possible explanations for such communities by studying mixing patterns of SourceForge projects.

Community Structure of OSS Project Network

Detecting network communities is a difficult problem due to the high complexity of graph computation. Many algorithms have been designed to approximate the computation. A recent algorithm designed by Girvan and Newman [newman2 et al. 2004, Girvan et al. 2002] is based on edge betweenness. This method focuses on edges which are most important to the connection of the whole network. By progressively removing edges with the highest betweenness, groups can be separated from the whole network and reflect the underlying structure. However, this algorithm runs in $O(m^2n)$ time on a network with m edges and n vertices. On a sparse graph where $m \sim n$, the algorithm runs in $O(n^3)$. Such high complexity makes computation of large networks impractical. A faster algorithm is proposed to detect network communities by greedy algorithm [Newman 2004]. Unlike the previous top-down algorithm, this algorithm works in a bottom-up way: at the starting point, each node is a group; at each step, two communities are picked to join based on some measurements which results in the best grouping. This algorithm runs in time $O((m+n)n)$ on a random network, or $O(n^2)$ on a sparse graph. An improved method is presented by Clauset et al. [Clauset et al. 2004], which performs the same optimization, but implements more sophisticated data structure. This algorithm can run in an approximately linear time, $O(n \log(n^2))$.

We use the algorithm proposed by Clauset [Clauset et al. 2004] to analyze the community structure of OSS. This method is based on the greedy optimization of the quantity known as *modularity*, which measures if the division of a community is meaningful, that is, after the division, there should be many edges within communities and a few edges between them. According to Newman [Newman 2004], the *modularity*, Q , is defined as the fraction of edges within communities subtracts the expected value of the same quantity if edges fall in a random network. The computation of Q can be gotten by equation (1) and (2).

$$Q = \sum_i (e_{ii} - a_i^2) \quad (1), \quad a_i = \frac{k_i}{\sum_j e_{ij}} \quad (2)$$

Where, e_{ij} is the fraction of edges that connect nodes in group i to group j , a_i is the fraction of edges that connect to nodes in group i versus the total edges in the network, and k_i is the number of edges connecting to group i . The method uses a greedy optimization to repeatedly group two communities whose amalgamation results the largest increase in Q . And the best community structure is at where Q is the largest.

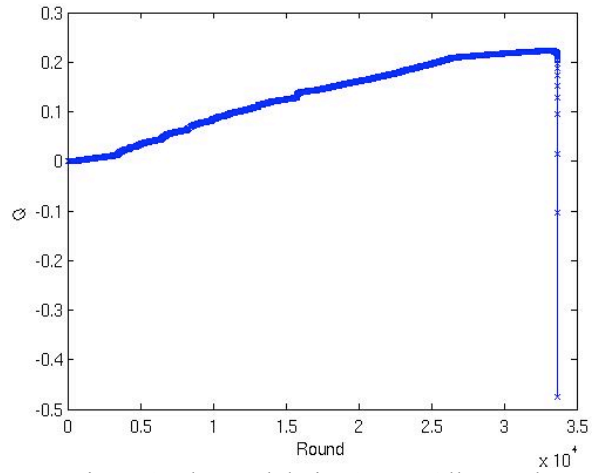


Figure 1. The Modularity Q over All Rounds

We studied the OSS project network in SourceForge. In this network, two projects are connected if they have one or more common developers. We applied the algorithm to the project network to detect communities. We examined the largest component of the project network in January 2003. We also exclude project SourceForge [Sourceforge] because it has links to over *10,000* other projects. Thus, the project network we studied consists of *27,834* nodes and *173,644* edges.

Figure 1 gives the value of modularity Q over all rounds. The highest value of the modularity is $Q = 0.2227$, which occurs when there are *611* groups. It contains several large communities and many small communities. The largest group consists of *3467* projects and there are many groups of size less than *10*. Figure 2 shows the distribution of the group size. We can observe the group size follows a power law distribution. The slope of its log transformation is *0.9432*. Table 1 gives the *10* largest groups in this network. These *10* largest groups include *68.4%* of the whole SourceForge projects, while the rest *601* groups only take about *30%* of all projects.

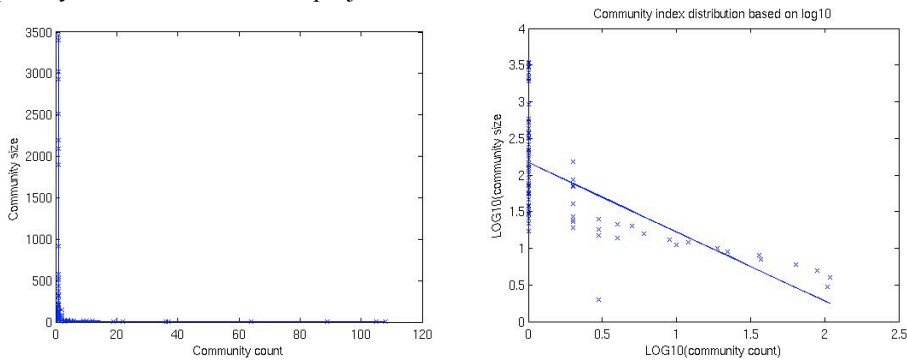


Figure 2. Community Degree Distribution of the Largest Component of the Project Network

Our results also provide information about communications and knowledge flow about the project networks. The important communication paths are those connections between communities. According to their interests, some developers participated on some projects which target to different topics and belong to different communities. These developers become keys to transfer information between two seemingly unrelated project groups.

Table 1. The 10 Largest Communities in the Project Network

Community size	Number of such communities
3467	1
3404	1

3023	1
2931	1
2511	1
2198	1
2096	1
1898	1
911	1
576	1

OSS Assortative Mixing

Several explanations exist for the formation of the community structure. Communities can be the result of the mutual acquaintance mechanism [Jin et al. 2001], that is, two nodes with a common neighbor are more likely to link to each other than those without common neighbors. This mechanism has been proved to work in social networks [Jin et al. 2001]. However, in many social networks, communities cannot simply explained by mutual acquaintances. Communities may be present for nodes with the same attributes. Thus, homophily plays an important role for the community structure. For example, when we study communities of people, we may consider their ages, languages, and nationalities. Assortative mixing [Newman et al.2003] is a way to study the preferential association of nodes to others that are like them in some way.

The level of assortative mixing can be measured by an *assortative coefficient*, which is defined as:

$$r = \frac{\sum_i e_{ii} - \sum_i a_i b_i}{1 - \sum_i a_i b_i} \quad (3)$$

where e_{ij} is the fraction of edges in a network which connect nodes of type i to nodes of type j , a_i and b_j are the fraction of edges of each type which is attached to nodes of type i . They satisfy the following rules:

$$\sum_{ij} e_{ij} = 1 \quad (4) \quad \sum_j e_{ij} = a_i \quad (5) \quad \sum_i e_{ij} = b_j \quad (6)$$

Sourceforge provides several categories for hosted projects. A project can be classified to its topic, operating system, user interface, development status, intended audience, and programming language. Each category contains hierarchical subcategories. In our study, we only consider the first level subcategories.

In Sourceforge, a project might be listed into multiple subcategories in those categories. For such kind of projects, we treat them as a separate project in each subcategory. This creates multiple edges with different kinds for a pair of projects.

We calculated the assortative mixing coefficient for each category based on equation (3). Table 2 shows Sourceforge categories, their corresponding first-level subcategories, and the assortative coefficient for each category. There are about 30% projects on SourceForge which are not assigned to a category. We exclude these projects when we calculate assortative coefficients. Assortative coefficients for all categories are positive, which means projects in the same categories prefer to associate with each other. Among those categories, programming language, operating system, and topic have strong assortative mixing effects. Because our project network is linked by people who join multiple projects, those strong assortative mixing coefficients can be explained by the fact that when an existing developer makes a decision to join a project, he/she tends to join projects which have the same programming language, operating system and topic as his/her current projects.

Conclusions

In this paper, we conduct the identification of the community structure for the SourceForge project network. This network is constructed by common developers among Sourceforge projects. We found that groups exist in the SourceForge project network. Furthermore, we explore possible reasons for the formation of those groups by examining assortative mixing coefficients for projects categories. Among them, we found projects with the same programming languages, operating systems and topics are more likely to be grouped together. Our research provides useful information to study the interaction between projects and the communication and information flow in OSS virtual organization.

Category	First-level Subcategories and Their Percentage	r
----------	--	---

Topic	Communications(7.7%), security(1.3%), development(8.1%), desktop(1.8%), editors(1.3%), database(3.0%), education(1.5%), games(7.3%), internet(12.0%), scientific(3.9%), multimedia(5.9%), office(2.3%), religion(0.1%), system(9.7%), printing(0.2%), terminals(0.3%), other(1.2%), unknown(32.2%)	0.1009
Operating System	Posix(25.2%), Microsoft(14.0%), os2(0.1%), macos(1.9%), beos(0.4%), independent(14.7%), pdasystems(0.5%), other(0.8%), unknown(33.7%)	0.1078
User Interface	Console(10.6%), x11(10.2%), win32(9.3%), web(11.7%), daemon(3.5%), cocoa(0.7%), handhelds(0.4%), other(4.7%), unknown(36.3%)	0.0893
Development Status	Planning(12.8%), prealpha(8.6%), alpha(7.7%), beta(9.1%), production(7.0%), mature(7.3%), inactive(0.1%), unknown(31.7%)	0.0553
Intended Audience	End users(21.9%), developers(24.9%), system administrators(10.7%), customer service(0.2%), education(0.8%), financial insurance(0.09%), health care industry(0.07%), information technology(1.0%), legal industry(0.4%), manufacturing(0.9%), religion(0.4%), science research(0.7%), telecommunications(0.2%), other(5.4%), unknown(32.6%)	0.0449
Programming Language	C(11.1%), c++(10.6%), java(8.4%), php(6.4%), perl(4.7%), python(2.4%), visual basic(1.3%), other(0.6%), Unknown(32.9%)	0.1541

Table 2. Projects Categories and Subcategories

References

- [Clauset et al. 2004] Clauset A. & Newman M., "Finding community structure in very large networks", *Physics Review, E*, 70(066111), 2004.
- [Girvan et al. 2002] Girvan M. & Newman M., "Community structure in social and biological networks", *Natl. Acad. Sci. USA*, 2002.
- [Jin et al. 2001] Jin E. M., Girvan M., & Newman M., "The structure of growing social networks", *Phys. Rev. E* 64. 046132. 2001.
- [Lopez et al. 2004] Lopez-Fernandez, L., Robles, G. & Gonzalez-Barahona, J.M., "Applying social network analysis to the information in CVS repositories", *Proceedings of the First International Workshop on Mining Software Repositories (MSR 2004)*, Edinburgh, UK, 2004.
- [Newman 2004] Newman M., "Fast algorithm for detecting community structure in networks", *physics Review, E* 69(066133), 2004.
- [Newman2 et al. 2004] Newman M. & Girvan M., "Finding and evaluating community structure in networks", *Physics Review, E* 69(026113), 2004.
- [Tyler et al. 2003] Tyler J. R., Wilkinson D.M., & Huberman B. A., "Email as spectroscopy: automated discovery of community structure within organizations", *Communities and technologies*, (isbn 1-4020-1611-5), published by kluwer, B.V. 81-96, 2003.
- [Wukjubsib et al. 2004] Wilkinson D. & Huberman B. "A method for finding communities of related genes", *Natl. Acad. Sci. USA*, 2004.
- [Madey et al. 2004] Madey G., Freeh V., and Tynan R., "Modeling the F/OSS Community: A Quantitative Investigation," in *Free/Open Source Software Development*, ed., Stephan Koch, Idea Publishing, 2004.
- [Sourceforge] <http://www.sourceforge.net>
- [Xu et al. 2004] Xu J. & Madey G., "Exploration of the Open Source Software Community", *NAACSOS Conference 2004*, Pittsburgh, PA, June 2004
- [Xu1 et al. 2003] Xu J., Huang Y., Madey G., "A Research Support System Framework for Web Data Mining", *Workshop on Applications, Products and Services of Web-based Support Systems at the Joint International Conference on Web Intelligence (2003 IEEE/WIC) and Intelligent Agent Technology*, Halifax, Canada, October 2003.
- [Xu2 et al. 2003] Xu J., Gao Y., Goett J., Madey G., "A Multi-Model Docking Experiment of Dynamic Social Network Simulations", *Agents2003*, Chicago, IL, October 2003