Understanding a Developer Social Network and its Evolution[*]

Qiaona Hong, Sunghun Kim, S.C. Cheung
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{qiaona, hunkim, scc}@cse.ust.hk

Christian Bird
Microsoft Research
cbird@microsoft.com

***Abstract*—With the growing number of large scale software projects, software development and maintenance demands the participation of larger groups. Having a thorough understanding of the group of developers is critical for improving development and maintenance quality and reducing cost. In contrast to most commercial software endeavors, developers in open source software (OSS) projects enjoy more freedom to organize and contribute to a project in their own working style. Their interactions through various means in the project generate a latent developer social network (DSN). We have observed that developers and their relationships in these DSNs change continually under the influence of differences in the set of active developers and their changing activities. Revealing and understanding the structure and evolution of these social networks as well as their similarities and differences from other more general social networks (GSNs) is of value to our software engineering community, as it allows us to begin building an understanding of how well the findings from other fields based on GSNs apply to DSN. In this paper, we compare DSNs with popular GSNs such as Facebook, Twitter, Cyworld (a large social network in South Korea), and the Amazon recommendation network. We found, for instance, that while most social networks exhibit power law degree distributions, our DSNs do not. In addition, we also examine how DSNs evolve over time, highlighting how events within a project (such as a release of new software or the departure of prominent developers) impact the makeup of the DSNs, and observe the evolution of topological properties such as modularity and the paths of communities within these networks.**

Keywords-developer social network;community detection

I. INTRODUCTION

Due to the great increase in the scale of software projects recently, software development and maintenance are highly social activities which have attracted study from many researchers [10, 19, 15]. As developers work together on software projects, they form implicit collaborative social networks [19]. Some researchers have begun to apply ideas from general social networks (GSNs) such as Facebook and twitter to these developer social networks (DSNs). For example, Begel et al. [19] proposed a social networking web service, codebook, containing pages with information about developers and artifacts in a style similar to Facebook. GSNs such as Facebook, Twitter, and Cyworld have enjoyed much success as they have leveraged and enhanced social relationships. As researchers study developer collaboration networks, a natural question to ask is, "How similar are developer social networks to general social networks?" As both DSNs and GSNs undergo more study, it is useful to know how similar DSNs are to GSNs.

Unlike general social networks (GSNs), a developer social network (DSN) is an underlying network which only reveals itself after being extracted from an open source project. Developer social networks generally have stricter control over topics than GSNs. In GSNs such as Twitter and Facebook, participants are encouraged to post on any topic and express their views or feelings. However, developers in DSNs are restricted to contribute on project-related topics. Meaningless or unrelated opinions can be grounds for being banned from the project. Therefore, we expect that DSNs will differ from GSNs in certain aspects. In contrast, developers are similar to participants in GSNs in some respects. Neither is forced to participate.

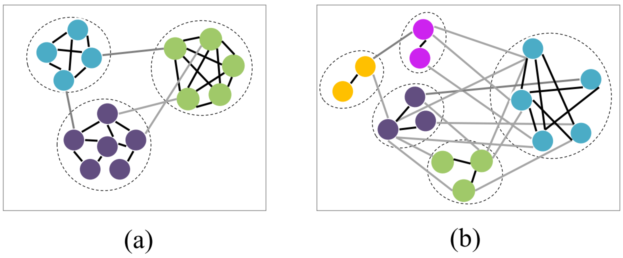

Figure 1. An example of community structure in networks.

Much research exists on GSNs including Facebook [7], Twitter [6], and Cyworld [5], and there is pre-existing work examining DSNs in OSS projects [10, 15, 16]. However, since there has been no formal comparison of these two types of networks, it is unclear which properties are general and what is specific to these individual types of social networks. This paper presents a detailed comparison between DSNs and GSNs from multiple aspects including community structure.

Community structure in social networks which has attracted considerable attention is an important aspect to consider when making the comparison between DSNs and GSNs. In 2002, Girvan and Newman [20] introduced the notion of community structure within networks as the division of a graph into subgraphs (called communities) in which the connections within the subgraphs are much denser than the connections between them. They also defined a metric, termed modularity [9], for evaluating how well a graph can be partitioned into these communities. A graph with high modularity has very well defined and tight

[*]This research is partially supported by a grant from the Research Grants Council of Hong Kong (Project No. 612108).

TABLE I. INFORMATION OF SUBJECTS

Subject	Dec. 2009	Oct. 2009 ~ Dec. 2009	Jul. 2009~ Dec.2009	Jan. 2009~ Dec. 2009	Jan. 2008~ Dec. 2009	Jan. 2006~ Dec. 2009
Time Period	1 month	3 months	6 months	1 year	2 years	4 years
# of developers (remained/total)	387 /2996	558 /6046	836 /10290	1226 /16318	1950 /31557	2865 /51692
# of bug reports	10586	32223	56904	95721	160626	246331
# of comments	37056	131569	267749	511777	968339	1610999

knit communities with few connections between them, while a low modularity graph has poor structure. The networks in Fig. 1 have the same amount of nodes and edges. However they have very different structures. The network in Fig. 1(a) displays strong community structure and has a high modularity 0.51 while the network in Fig. 1(b) has weak community structure. There is not much difference between the densities of inter connections and intra connections in this network. It has a low modularity of 0.176. Identifying the optimal partition of nodes into communities is an NP-complete problem, but in the recent years much progress has been made in developing algorithms that work well in practice. Beyond this comparison we also provide a detailed study of DSNs themselves in terms of community evolution, providing a reference for researchers and developers. This study addresses several questions. We list them here.

- Do DSNs display similar characteristics to GSNs? Which characteristic are distinct?
- How does a DSN evolve over time?
- Does a DSN have significant community structure, evidenced by tightly knit communities within a project?
- What is the evolution of these communities in a DSN? Are there obvious patterns or trends evident in this evolution? Does the observed evolution correspond to key events in the project lifetime?

To address the above questions, we begin with extracting DSNs. We take developers involved in the Mozilla Bug Tracking system[1] as subjects of analysis since Mozilla is among the most popular and mature open source projects. In the DSNs, the nodes represent developers and the edges correspond to developers' connections via shared bugs.

The structure of this paper is as follows. Section 2 describes our approach for extracting and identifying the communities in DSNs. In Section 3, we study the topological characteristics of the developer social network. Section 4 presents the topological characteristic evolution of the DSN, and Section 5 investigates community evolution. Section 6 discusses some threats to validity, and Section 7 surveys related work. Finally, we conclude all lessons learned in Section 8.

II. DEVELOPER SOCIAL NETWORK COMMUNITIES

This section describes the methodology of extracting DSNs from Mozilla bug reports and their comments from 2000 to 2009. From the DSNs, we identify communities, which are the basic units of our study in community evolution. We employ a well-known and widely used community identification algorithm, Louvain [11]. Finally, we evaluate our methodology by comparing identified communities with off-line birds-of-a-feather (BOF) meetings at the 2010 Mozilla Summit[2].

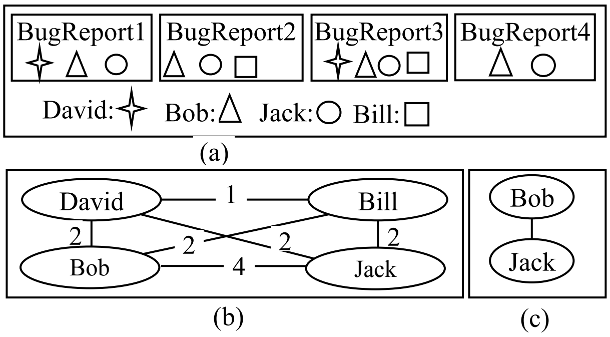

Figure 2. An illustrative example of the approach for extracting DSNs from a bug tracking system.

A. Extracting Developer Social Network

Bird et al. examined the social interactions on developer mailing lists of Apache [16]. Xu et al. examined co-membership in SourceForge projects [22]. Each of these networks is based on a particular form of relationship. However, we choose to examine DSNs in the context of bug tracking systems for two reasons. First, the majority of bugs are short lived, indicating a strong temporal relationship. Previous work has examined collaboration on source files, but using project membership or file contributions may indicate false relationships because both are long lived and the interactions may be distant temporally. Second, bugs are generally focused on one technical topic and fixed in a localized portion of the codebase. Bird et al. found that mailing list discussions can include general topics such as process discussions that large majorities of the community participate in. As our interest is modeling how developers work together, these discussions may represent noise in the data. Individual bugs do not elicit comments from a large proportion of the developer base as some mailing list discussions do, and require more effort than simply replying to a message.

Specifically, we extract the DSN by mining Mozilla bug reports and their comments. Since Mozilla developers are free to comment on any bug, the bug comments reflect developers' interest. Our underlying assumption is that developers who share the same interest are related to each other. We express their relationships in an undirected graph.

As an illustrative example, suppose we have three bug reports and four developers who commented on each bug report as shown in Fig. 2(a). Each symbol in a bug report indicates a comment in that bug report from the corresponding developer. Based on their comment

[1] http://bugzilla.mozilla.org
[2] http://wiki.mozilla.org/Summit2010/Meetings

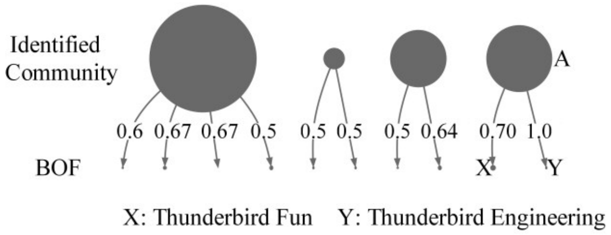

Figure 3. Mapping between identified communities and BOF meetings. Radius is proportional to number of developers.

relationships, we extract a developer network as shown in Fig. 2(b) indicating that developers who made comments on the same bugs are connected. We create weighted edges between developers by assigning a weight to each edge equal to the number of bug reports that the two developers have both commented on.

In the majority of open source projects (including Mozilla), bug reports are open to the public to encourage all users to report bugs and comment on the bugs. As a result, our developer network includes a large number of users who contributed to bug reports only a few times. While these contributions are of value to the community, we are most interested in examining the interactions of the core participants in the community - developers with direct code access and participants with a consistent history of activity working on bugs. To remove the "casual" users, we eliminate edges with a weight of only one or two. After this elimination process of edges, all nodes that no longer have any connected edges are removed from the graph. For example, after removing edges whose weight is equal or less than three and the resultant unconnected nodes in Fig. 2(b), we are left with the unweighted graph shown in Fig. 2(c).

We extracted 26 DSNs from 496,692 bug reports and 3,893,025 comments made by 106,123 developers in total. Six DSNs are extracted from different lengths of time, and the other twenty are extracted from a series of subsequent six month periods for examining DSN evolution. We provide detailed subject information analyzed regarding the different lengths of use in this paper in Table I.

B. *Identifying Communities*

After extracting DSNs from bug reports, we identify communities. Understanding community structure is critical to understand the network [10] as it allows us to identify sets of developers who share the same interests and work on similar issues. To identify communities, we use the Louvain [11] algorithm which is widely used in the social network analysis literature [1]. Kwak et al. found that Louvain outperforms other community detection algorithms on most subjects [1]. The Louvain method is probabilistic and produces slightly different values of modularity for the same graph as the input ordering of nodes change [1]. To mitigate this issue, we generated 50 sets of the same data with randomly perturbed input orderings of nodes for every network and present all the results.

C. *Evaluation of Identified Communities*

Once we have identified the communities in the DSNs, we need to verify that they reflect real divisions of developers into their communities. We validated the results in two ways.

First, we selected the developers with the highest node degree in the DSNs and examined the project to determine if these were leaders within the project. For example, we found that Gervase Markham, one of the highest degree nodes in our network, started to contribute to the Mozilla project in 1999 and is a leading developer in the Bugzilla project. Another example is Mike Beltzner, who is a famous Mozilla hacker. Gavin Sharp, another high degree node, is currently one of the most active developers in the Mozilla project. We verified 100 nodes in the network and found that all represented key Mozilla developers (they all made a large number of commits to the system and/or had a long history with the project, most spanning multiple years), validating that our DSNs do reflect reality.

Next, we evaluated the similarity between the identified communities in our DSNs and real offline developer meetings. We used the birds-of-a-feather (BOF) meetings in the Mozilla Summit 2010. BOF meetings represent communities in reality since developers that take part in those meetings are those that have an interest in them. Since these meetings were held only last year (2010), we identified developer communities in a DSN created from recent bug reports (Jul. 2009 to Dec. 2009).

The divisions on the top of Fig. 3 represent identified communities, and the divisions of the developers on the bottom represent BOF meetings. We measured how many developers overlap between real BOF meetings and identified communities. For example, community A in Fig. 3 consists of 70% of BOF X (the Thunderbird Fun/Product/Participation meeting), and 100% of BOF Y (the Thunderbird engineering/dev-process working session meeting). All of the developers from these two BOF meetings were placed in the same community by the Louvain algorithm. Since in Mozilla Summit 2010, they held many small BOF meetings, our identified communities include more than one BOF meeting. However the vast majority of BOF meetings belong to one community. This indicates that while BOF meetings may indicate a finer division of developers than our communities, our method rarely divides known groups of developers. We conclude that our identified communities reflect real groups of developers with similar interests and concerns. Furthermore, we consulted a Mozilla developer, Channy Yun, about our identified communities, and he confirmed that they reflect real Mozilla developer structures.

III. DEVELOPER SOCIAL NETWORK VS GENERAL SOCIAL NETWORK

This section compares DSNs to various GSNs in various domains including Facebook, Cyworld, and Twitter by measuring commonly used social network metrics such as Power Law, Degree of Separation, Modularity, and Community Size. Basic information of all used GSNs are shown in Table II.

In addition, we compare DSNs extracted from different lengths of time, which include the most recent 1 month, 3 months, 6 months, 1 year, 2 years, 4 years (all ending in Dec. 2009) For brevity, we name DSNs extracted from different periods as length-DSN (e.g. 1-month DSN, 3-month DSN, … , 4-year DSN).

A. *Power Law*

Various networks (e.g., WWW [18], social [6]) display a power-law node degree distribution, having only a few nodes with very high degree and a large number of nodes with low degree. When plotted on a log-log plot, a power law generally follows a straight line. This property of networks is indicative of the existence of a small number of "hubs" in the network that act as influential nodes and

TABLE III. INFORMATION OF GSNs

	WWE	Twitter	Cyworld[5]	Cyworld[1]	Facebook	Amazon	FL	GL	Hugged
# of nodes	143736	87897	12048186	11537961	63730	409687	16800000	617864	116376
# of edges	707761	829247	190589667	177566730	817090	2464630	73300	277540	51343

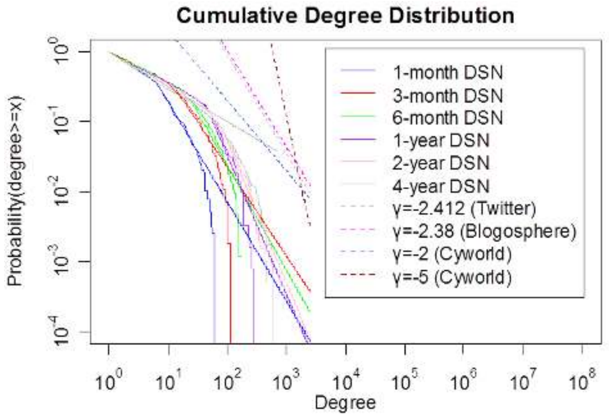

Figure 4. Topological characteristics of DSN. The y-axis presents the complementary cumulative distribution function (CCDF), and the x-axis presents the degree of a node.

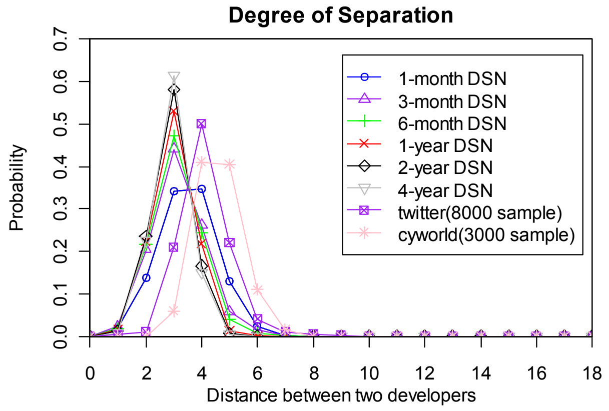

Figure 5. Degree of Separation for various DSNs and GSNs.

information brokers and a large number of peripheral members with few connections [23].

We start our comparison of DSNs with GSNs by investigating the degree distribution. Most GSNs have a power law degree distribution with an exponent, r, between -2 and -3 [6]. For example, the exponent for blogosphere is -2.38 for the Weblogging Ecosystems Workshop collection [4] which attests the existence of key bloggers who have a high number of blogging friends. This also holds true for Twitter which has a power law distribution in both in- and out-degree with the same exponent -2.4. What is more interesting is that some networks like Cyworld, a famous large scale South Korean social network service, show two different scaling regions, a rapid decay (r~-5) and a heavy tail (r~-2) [5]. We applied the approach of analyzing power law distributed data introduced by Clauset et al [24] to obtain the power law distribution exponent for each DSN. Based on the visualization in Fig. 4, we found that only a small portion of the curve can be fit to a power-law distribution. Therefore, we conducted the quantitative power law fit test introduced by Clauset et al. [24] to test whether the DSN degree distribution is different from a power law distribution to a statistically significant degree. The p-value, which is the likelihood that the DSN degree distribution actually does follow a power-law (the null hypothesis), was less than 0.1 for all DSNs in Fig. 4, indicating that none follow a power law distribution.

We therefore conclude that different from GSNs, DSNs do not have a power law degree distribution, irrespective of length of time. However, the degree distributions in DSNs have some properties similar to those in GSNs. DSNs also have a large portion of developers with low degree and a small portion of developers with high degree. Moreover the portion of high degree nodes is relative small in DSNs.

B. Degree of Separation

Degree of separation, the shortest distance between any two nodes of a network, has become a crucial metric for analyzing the social structure ever since Stanley Milgram reported the famous “six-degrees of separation” experiment in 1969 [12]. With the emergence of social networks, degree of separation has been well studied in various social networks including Cyworld and Twitter [5, 6]. The total number of registered users in Cyworld was 12 million as of November 2005, when Ahn et al. reported that the average path length between 90% of nodes in Cyworld was less than 6 [5]. Kwak et al. found that the average path length of Twitter is 4.12 [6]. For 70.5% of node pairs, the path length is 4 or shorter and for 97.6% it is 6 or shorter.

Since the scale of DSNs is smaller than that of GSNs, we employed the Floyd-Warshall algorithm [13] to obtain the distribution of the shortest path length between any two nodes. Compared to the Breadth-First algorithm which is used in the Twitter and Cyworld analyses, the Floyd-Warshall algorithm is more efficient for smaller scale networks.

The average path length in DSNs varies from 2.9 to 3.4. The developers in 1-month DSN are farthest from each other. For 91.9 % of pairs of nodes in it, the path length is 5 or shorter. In the rest of the DSNs, the path length is 4 or shorter for at least 90% of pairs of nodes. In addition, we see a slight downward trend in the average path length when the length of period increases. This result is not surprising. Although there are additional developers, there will also be many more connections in DSNs over time. Overall, the developers in DSNs are closer to each other than participants in GSNs such as Twitter and Cyworld.

Like GSNs, DSNs also have the so-called “small world” property which means most pairs of developers are connected within a few hops in the same way as pairs of participants’ in GSNs. Moreover, the average path length in DSNs is much shorter than that of Cyworld and Twitter. We speculate that this might be due to the fact that users on Cyworld and Twitter have a wider choice of topics while developers in DSN are restricted to participate in a narrower range of topics, therefore increasing the likelihood of shared interests.

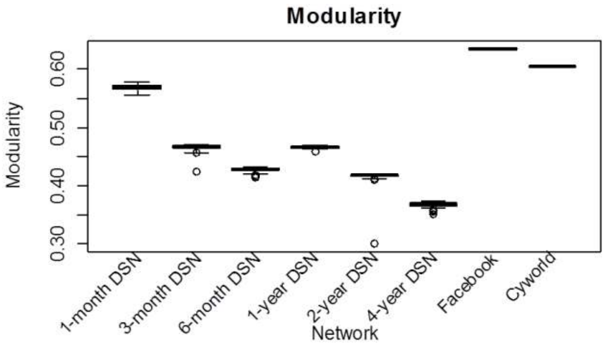

Figure 6. Modularity for DSNs of different time durations

C. Modularity

Modularity, Q, is the standard measure [1, 3, 10, 11] used to quantify the strength of a community structure. Higher modularity indicates that there are clearly defined communities (teams) within the network. Modularity was introduced by Newman and Girvan in their original community structure paper [9] and for some partition of a graph into communities, its definition is:

$$Q = \sum_i (e_{ii} - a_i^2) \qquad (1)$$

where e_{ii} is the proportion of the edges between nodes within community i over all edges in the graph, and a_i is the proportion of all edges that cross the community boundary. When the value of modularity is 0, the community structure is no stronger than that of a randomly generated network; there are no communities within the network. The Maximum value of modularity is 1. We employ modularity in an effort to understand the community structure of DSNs more easily. In our context, the modularity indicates if there is a clear division of the Mozilla developer population into teams or if the project is fairly integrated with developers coordinating on bugs in an unorganized fashion.

Kwak et al. [1] obtained the modularity of 12 social networks including Facebook and Cyworld, showing that except for the now famous Zachary's Karate Club (a small network that has become a de facto benchmark in community structure work), social networks have significant community structure since their values of modularity are all above 0.3 [6]. In practice, 0.3 is a threshold above which a naturally occurring network is said to be highly modular [9].

In this section, we examine whether DSNs have as strong a community structure as other GSNs by applying the Louvain algorithm on all DSNs in Fig. 6. As mentioned in Section 2.2, Louvain produces slightly different results on the same network with different input orderings of nodes, so we examined the results for 50 different orderings for each subject DSN. Overall, modularity values are always above 0.3 for the DSNs. With the length of time period increasing, the modularity shows a fluctuant decrease. The highest modularity was obtained in the 1-month DSN, where the median value was 0.57.

We conclude that similar to GSNs, DSNs have significant community structure, as reflected by their modularity.

D. Community Size

Within a network with community structure (i.e. high modularity), the community size refers to the total number of nodes within a community. The community size is a key quantitative characteristic of community structure in a network as it indicates if communities are disparate or uniform in their division of a network [17].

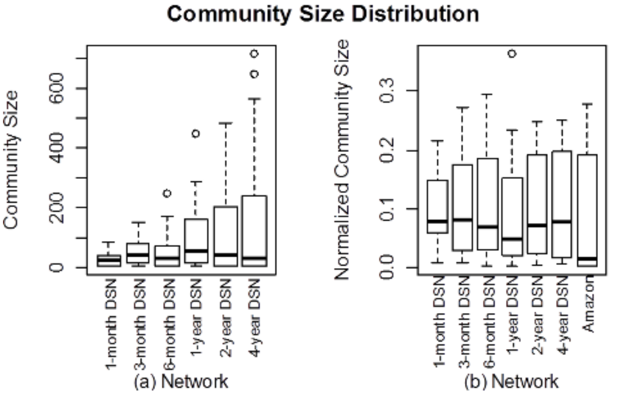

Figure 7. Community size distribution. Normalized community size is measured in terms of proportions of all nodes in the graph since Amazon is much larger.

Nazir et al. [7] compared the community sizes of three game applications in Facebook: Fighters' Club (FC), Get Love (GL), and Hugged. All three games have more than 10,000 users. The biggest community for FC accounts for 72.6% of the total users, while the biggest communities in GL and Hugged account for less than 10% of the users. They also found that FC has a biased distribution of community size, but communities from both GL and Hugged have similar wider community size spreads. Similarly, Clauset et al. [8] studied the community size in the Amazon.com network which has more than 400,000 users. The biggest community accounted for 28% users of the entire network. They found that the community size distribution of Amazon.com network follows a power law. Following a similar methodology, we measured the community sizes of DSNs in different length of periods ranging from 1 month to 4 years. Again, we ran the Louvain algorithm 50 times dividing the DSNs into sub-communities and computed the sizes of communities in the division for each DSN.

Fig. 7(a) shows the inter-quartile box-plots of the community size distributions. The sizes of the communities are small regardless of the length of time period. For instance, the biggest communities in 1-month and 4-year DSNs only have 83 and 718 developers respectively. The biggest community of the DSNs accounts for 21% ~ 36% of the users and the ten biggest communities account for more than 99% of the users for all DSNs. The community size median is small and varies from 23 to 55. The distributions of community size display a similar property to a power law distribution, with a few big communities but many small communities.

A DSN has many similar characteristics to a GSN in terms of community size. First, several big communities account for almost all users in the network. The ten biggest communities account for 87% of users in Amazon.com while that accounts even more in DSNs. Second, the biggest community accounts for similar percentage of users, 28% in Amazon, 21%~36% in DSNs. Fig. 7(b) plots the size distribution of the ten biggest communities after normalizing by the total size of the network. The distributions are surprisingly similar despite the fact that they differ in size by orders of magnitude.

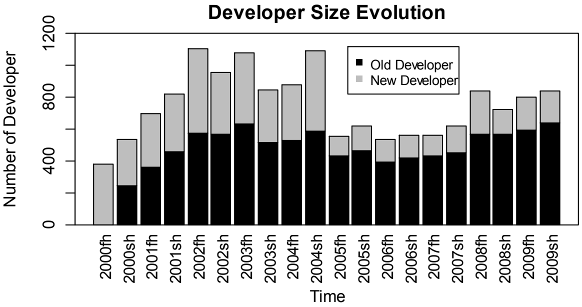

Figure 8. Number of active developers over time

E. *Analysis Summary*

We summarize the DSN analysis results.

- Unlike GSNs, DSNs do not follow a power law degree distribution. However, DSNs do have other similar properties to GSNs such as having a large portion of nodes with low degree and only a small portion of nodes with high degree.
- Similar to GSNs, DSNs also have the small world property of low degree of separation. However, DSNs have a "smaller" world property.
- DSNs and GSNs both have strong community structure, with modularity values above 0.3.
- The size of communities in DSNs is small compared to that in most GSNs. We also find that DSNs have a widespread community size distribution and their biggest community accounts for 21% to 36% of the total developers.
- Regardless of time period (except 1-month DSN), DSNs have very similar social network properties including degree of separation, modularity, and community size. In addition, we found that the 6-month DSN is a representative of all time durations beyond 6 months.

IV. DEVELOPER SOCIAL NETWORK EVOLUTION

We have characterized the differences and similarities of DSNs to GSNs. In this section, we turn to an examination of the evolution of social network properties over time. Here we are also only interested in participants with a consistent history of activity working on bugs.

A. *Identifying the Length of Unit Period*

The first step in observing DSN evolution is to decide the basic time period unit. We can observe DSN changes for every month, every year, etc. Based on our observations described in Section 3.5, we use 6 months as a representative period and observe how the DSN changes every 6 months. In our figures, we divide each year into first half of the year (using an "fh" suffix) and second half of the year ("sh").

B. *Developer Changes*

We first observe developer changes. Fig. 8 shows the number of new developers and old developers for every 6 month period. For each column, the gray bar indicates the number of developers that are new in that time period and black, the number of developers active in this time period that have been active in an earlier time period. The total number of developers falls into the range of [500, 1200] except for the first period (i.e., the first half year of 2000).

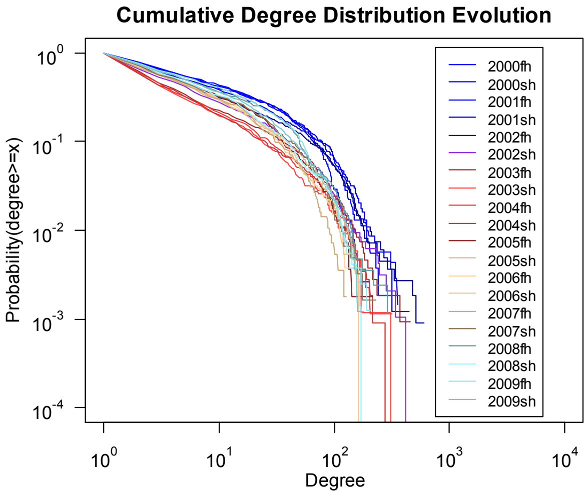

Figure 9. Degree distribution of DSNs over time

We also observe that the total number of developers increases until 2002. After that, there is a sharp fall-off around 2004 dropping from 1,093 developers to 533.

Fig. 8 also shows new developers continually joining the DSN. From 2000 to 2004, new developers account for more than 39% of the developers in the DSN. Fig, 8 shows a similar sharp fall in the percentage of new joining around 2004. We have found evidence that this is related to the Firefox 1.0 release in September 2004. A number of influential developers who developed Netscape/Mozilla left the community soon after. For example, Blake Ross, a core developer, began his new company after the release. Similarly, a few core developers including Ben Goodger and Katsuhiko Momoi joined Google in 2005. We define a "core" developer to be someone that has direct source code commit privileges, contributes non-trivial amounts of code to multiple parts of the system, and has been active in the community for a period of years. We note that these developers who had an influence on the community also had corresponding high degree in the DSN and were important members of their communities, indicating that these social network measures are indicative of real world impact.

From the above observation, we find that the behaviors of influential people often affect other developers in the DSN. The drastic drop in total number of developers in the DSN may be due to the departure of these influential developers.

C. *Degree Distribution Evolution*

We also observe the evolution of degree distribution since the degree distribution is a key property of a network (e.g. indicating if a small number of developers play a disproportionately important role). As we did in Section 3.1, we also tested the power law distribution hypotheses for all DSNs in Fig. 9. The p-values for 14 DSNs are less than 0.1 while for the other 6 DSNs the p-values are greater than 0.1, indicating that over two thirds of the DSNs differ from GSNs in that respect. For those which were not statistically different from a power-law, their large p-values might be due to lack of enough sample data [24]. Although DSNs do not display a power law degree distribution over time, they always have a large portion of nodes with low degree and a small proportion of nodes with high degree over time.

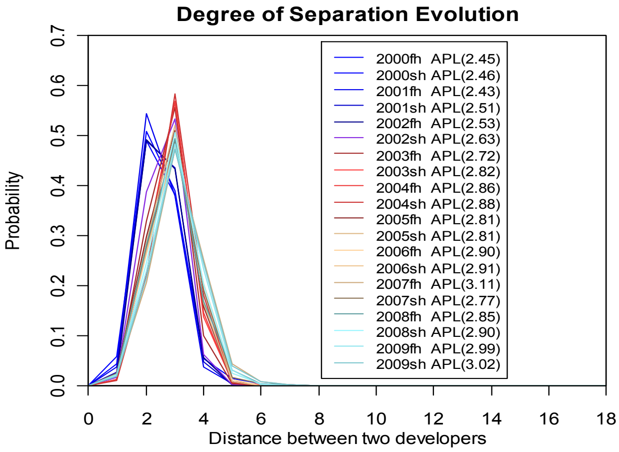

Figure 10. Evolution of Degree of Separation in Mozilla DSNs

D. *Degree of Separation Evolution*

Fig. 10 shows the degree of separation evolving over periods. Overall, the average path length increases slowly from 2.45 to 3.02 over the 10 years. Although the distance between developers increases a small amount over time, developers are always quite close to each other in the DSNs. More than 93% pairs of developers are within four hops of each other over the whole time period. We conjecture that the small increase is due to the large amount of newly incoming bug reports (e.g., on average, 24,964 per period). Over time, the rate of bug reporting and commenting has increased. However, there is no such increase in the number of developers every period. The probability of two developers commenting on the same bug report at random decreases over time, which might lead to the increase of average path lengths. This is true for all but three periods: the first half year of 2001, 2005 and 2007. In conclusion, the increase in average path length is small and oscillates within a very narrow range above 2.7 after 2004. Thus, we consider it fairly constant over the long term (i.e., the average path length approaches a fairly stable level).

E. *Modularity Evolution*

We next examine modularity within the DSNs. Fig. 11 presents boxplots of the modularity for each period which consists of 20 experiments, each containing 50 runs with perturbed input node order. It displays a fluctuant increase in the modularity over periods from the first half year of 2000 to the second half year of 2009. The lowest modularity value is 0.20 (e.g. median of modularity values obtained from 2000sh-DSN) while the highest modularity value is 0.52 (e.g. median of modularity values obtained from 2009fh-DSN). All DSNs extracted before June 2003 have a modularity value below 0.3 indicating a more integrated community [8]. In contrast, the modularity values of all DSNs obtained after July 2003 are above 0.3 and we see a general upward trend over time, implying that the community has split into more well-defined teams over time.

We observe that the modularity increases over time. Since there is no mandated structure, the organization of the project and architecture of the code may be somewhat volatile as the project gradually and organically converges to a stable and accepted structure. Baldwin [21] found that the restructuring of the Netscape codebase (from which Mozilla originated) had both architectural (the code became more modular) and organizational (more newcomers began joining the project) impact afterwards. In addition, the Mozilla Foundation, which exists to support and provide leadership for Mozilla project, was officially launched on July 15, 2003. This likely contributes to the increase in modularity after 2003. Overall, we conclude that DSN modularity shows a clear increasing trend, indicating that teams are becoming more well-defined within the Mozilla project.

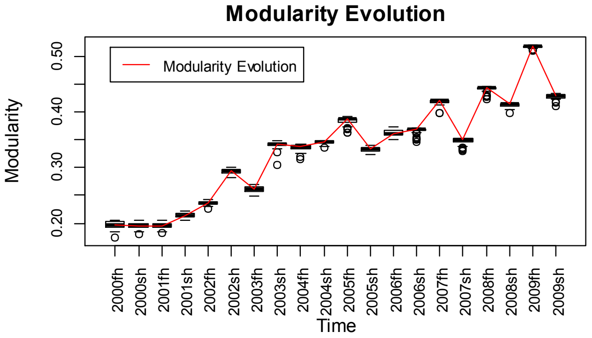

Figure 11. Modularity evolution

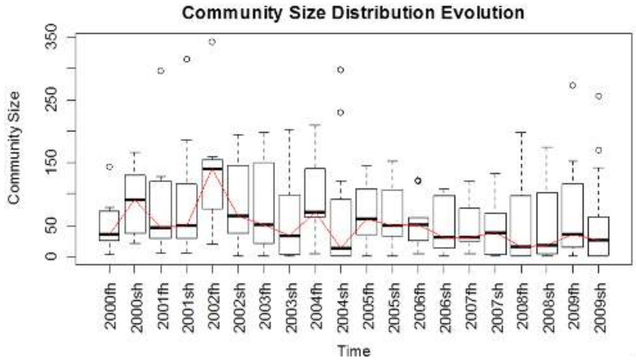

Figure 12. Evolution of community sizes in Mozilla over time

F. *Community Size Distribution Evolution*

Fig. 12 presents the community size distribution evolution over 10 years starting from 2000. The biggest community accounts for 20% - 43% of the developers in the DSNs, and the size of the median community falls into the range [14, 141] fluctuating over time instead of increasing or decreasing.

Based on this observation together with the fact that the total number of developers in DSNs for each period is less than 1,200, the community sizes in the DSNs are generally small (most below 75 members), with few changes over time. We see much more variance at the beginning of the project, when developer turnover was higher, than we do in the latter half of the project. We conclude that although community sizes initially were quite unstable, they have remained fairly consistent over the last 5 years. Again, this confirms that a team structure is apparent in the Mozilla project.

G. *Analysis Summary*

We summarize the conclusions from our evolution analysis below:

- The sharp change of the developer numbers around 2004 implies a big adjustment in Mozilla following the Firefox 1.0 release. Afterwards, the number of active developers is stable over time.

- DSNs have a large proportion of nodes with low degree and a very small proportion of nodes with high degree over time.
- In DSNs, the average path length slowly increases over time, but never grows to the size of GSNs, as over 90% of pairs of developers remain connected within three hops.
- The modularity of DSNs increase steadily over time, indicating communities that become more tight-knit.
- DSN communities are usually small without notable size change over time.

V. COMMUNITY EVOLUTION

In this section, we change our focus to examine how the communities within the Mozilla DSNs evolve over time.

A. *Community Evolution Patterns*

Lin et al. [2] proposed a community evolution taxonomy including five patterns: derivation, merge, split, extinct, and emerges. We refined these patterns and added interpretation of each pattern in the context of DSNs in table III. We identified all of these patterns in the community evolution for the DSNs as shown in Fig. 13.

TABLE III COMMUNITY EVOLUTION PATTERNS AND THEIR IMPLICATIONS ON DSNS.

Patterns	Description
Expand	Expanding is when a community increases in size and its prior community set comprises only one community. This indicates that newcomer developers are being attracted to this community.
Shrink	Shrinking is when a community decreases in size and its prior community set comprises only one community. This is evidence that the community's developers are leaving the project or they are joining other communities.
Merge	Merging describes when a community has at least two communities in its prior community set. This indicates at least two communities that have shared bugs and, therefore, common interests.
Split	Splitting is when a community has at least two communities in its post community set. This pattern shows that an interest discrepancy occurred in the single community.
Extinct	Extinction is when the post community set of a community contains no community. this implies that developers have left or completely scattered to a number of other communities
Emerge	Emergence is when the prior community set of a community contains no community. This may signify the emergence of a new interest or area of bugs.

B. *Evolution Map*

To observe community evolution, we need to trace communities over time. To this end, we use community similarity between two consecutive periods. Given a set of communities $C(p) = \{C_k(p)\}$ in time period p for $k = 1 \dots K$, where K is the number of communities in period p, we first compute the percentage of common nodes over the size of the smaller community for all possible pairs of communities $\left(C_k(p), C_k^*(p+1)\right)$. This is to determine which communities in the next time period are similar in makeup to each community in the current time period. Then we determine the closest community set $\{C_k^*(p+1)\}$ for each $C_k(p)$ by setting a threshold $\varepsilon = 0.3$ to filter out irrelevant communities. We refer to such a community set as a post community set of $C_k(p)$. Similarly, we refer the closest evolution community set $\{C_k^{*'}(p-1)\}$ as the prior community set of $C_k(p)$.

Each node in Fig. 13 represents a community, and each edge represents an evolution relationship between two communities. For example, an edge starting from community A and directed towards community B indicates that community B evolved from community A. In Fig. 13, communities detected in the same half year are aligned horizontally.

Overall, there are far more isolated nodes before 2005 than after, which indicates that more communities had weak stability before 2005. This might be due to lack of organization of developers. Firefox is arguably the most successful of the Mozilla projects and its 1.0 release was in September 2004. Mozilla Thunderbird made its 1.0 release soon after, in December 2004. This is consistent with our earlier developer evolution analysis, where we saw that developer turnover decreased dramatically after these releases. We examine several representative paths shown in Fig. 13.

Firefox (the bold_unfilled_circle path). The Firefox path in Fig. 13 shows the communities that contain several known Firefox developers including Gavin Sharp, Mike Beltzner, Ria Klaassen, and Mike Conner. These four Firefox developers eventually reached prominence and their node degrees were all among the top 10. They appear in communities along the path that of bold outlined circles, based on our conjecture that this path characterizes the activity of Firefox.

Bugzilla (the gray_filled_circle path). The Bugzilla path is long lived, evolving from 2001 to 2009. According to our observations, this path displays the evolution of the Bugzilla project. There is one short branch starting from the second half of 2008 which comprises the black filled circles. This short branch is related to the development of the Camino project.

Rhino (the dashed path). The sizes of all communities along this path are small. Based on the activities of developers in this path, we deduce that it is related to the development of the Rhino project. Norris Boyd, Rhino's creator, appears in most of these communities.

Security (the shadow_circle path). This path represents the evolution of the security community as many active developers in this path are members of the security group. There are a few branches that connect to this path, implying that developers in this group are also interested in other projects. This result is consistent with the fact that security is an issue affecting all projects.

We use several community evolution instances in Fig. 13 to illustrate whether the patterns observed reflect reality. The Expand instance labeled in the Firefox path coincides with the 3.0 release of Firefox on June 17, 2008. Many developers joined the Firefox community after this event through dealing with Firefox 3.0 bugs. The Shrink instance labeled in the Bugzilla path occurs at the same time as Bugzilla 2.16, the version that lasted the longest between releases. This high level of stability may have either caused or resulted from developers leaving. The Split instance labeled in the Security path is coincident with the release of Firefox 2.0. During this release, some security developers mainly focused on Firefox 2.0, while others stayed in the original security path. The subsequent Merge instance labeled in the Security path reflects the fact that after dealing with security problems in Firefox 2.0, those who focused on that codebase returned to their original community.

C. *Analysis Summary*

Some of the interesting findings made during our analysis include:

- The community evolution in the DSNs contains all five patterns found by Lin et al. [2].
- The community evolution in DSNs displays various paths which correspond to historical evolution of various sub projects in Mozilla and activity of core developers in Mozilla's history.
- At the beginning of an open source project the communities are fairly dynamic, but we observe that as time progresses, developers tend to "settle down" into fixed groups.

VI. Threats to Validity

Here we enumerate possible threats to validity in our study along with methods used to mitigate these threats.

In order to remove casual users, we removed nodes with edge weights of only one or two. We use this as indication that they do not have high levels of activity and are not key members of the community. While this means that we remove some members from our analysis, their low connectivity implies that these participants have little impact on the properties of DSNs that we have examined.

We also tried to determine a suitable length of time for observing the DSNs by inspecting four metrics: power law, degree of separation, modularity, and community size. While it is possible that different time periods may yield different results, we cover a comprehensive range of time periods.

Next, we examined some paths in the community evolution of the DSN and made the connection between these paths and both core developers and actual events by analyzing basic information of related bugs, checking information from Mozilla's website, and consulting developers in Mozilla. While it is possible that this evaluation may miss some key events, this method has been used frequently in the past with positive results [10].

In this work, we have only conducted analysis on the Mozilla project. We chose Mozilla because it is mature, large, considered successful, and has been well studied in prior research, allowing the reader to integrate our results with the findings of others. It is possible that these results on Mozilla may not generalize to other developer social networks. However, our methodology for analysis could easily be used with other projects such as Eclipse.

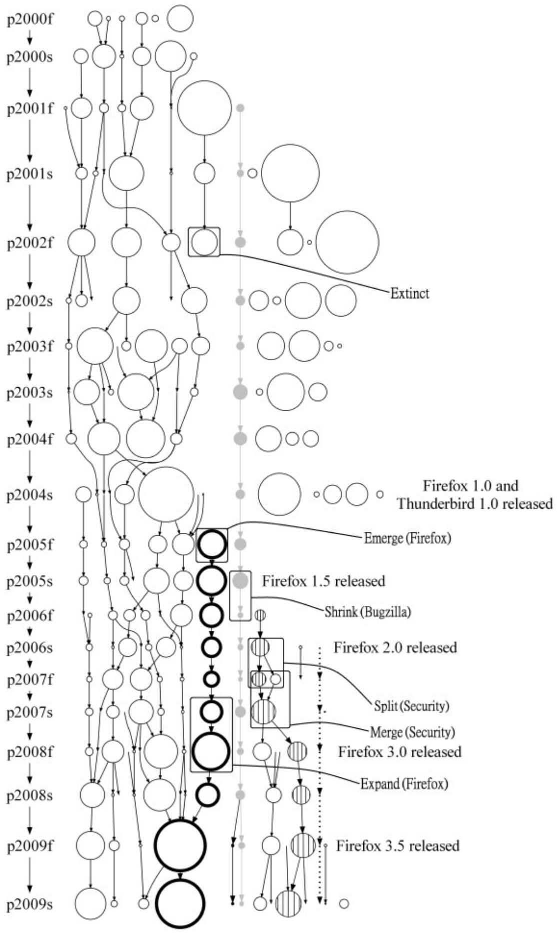

Figure 13. Community evolution path. The radius is proportional to the number of developers.

VII. RELATED WORK

There has been prior work on analyzing the static and dynamic properties of various networks. Due to space constraints, we survey the work related to our own.

Crowston and Howison [15] built social links between developers based on their co-occurrence information in bug reports. Their work aims to study the communication centralization problem in OSS teams. Our study also used the co-occurrence information of developers in bug reports to build social network among developers. However we aim to study both static and dynamic global properties of DSNs. Additionally the scale of the subject used in our work is much larger than theirs.

Bird et al. [16] extracted social networks from mailing list archives and empirically studied the differences between developers and non-developers from a social network metrics perspective. They also investigated the correlation between development activity and social network status of developers. In later work [10] they identified the community structure from the same social networks and demonstrated that their division of project was representative of the collaboration behavior of developers in OSS projects. Our work is complementary to this study by examining the community evolution patterns in DSNs and observing the individual community evolution paths. Additionally we conducted a study of the static properties of DSNs over periods of different lengths of time to in an effort to determine valid time durations for studying the dynamics of DSNs.

Lo et al. [5] extracted high-level statistics and detailed topological graph patterns from a developer collaboration network extracted from SourceForge.Net. Although we also study statistical properties of developer networks, we focus on investigating community structure and its evolution. We also conducted a comparison between DSNs and GSNs which is valuable when borrowing ideas from GSNs to apply on DSNs.

VIII. CONCLUSIONS AND FUTURE WORK

We have conducted a comprehensive analysis on DSNs based on a large project (in terms of both time and people). Our analysis on the static properties of DSNs indicates that DSNs extracted from a bug tracking system bear some resemblance to GSNs but also show key differences from them. In addition, our results give strong evidence of the community structure within DSNs. Second our study on the dynamic properties of DSNs shows that DSNs maintain small world characteristics over time. Our investigation of activity over the past 10 years indicates a gradual enhancement of community structure in DSNs. Furthermore, we observed the community evolution pattern in DSNs, examined the evolution paths of a number of projects within Mozilla, and found correspondence with key historical events in these projects.

Our study also provides new insights for developer social networking services like Codebook [19]. For example, our study found that the degree distribution of DSNs do not follow a power law. This should be taken into consideration when applying features that are affected by degree distribution such as "Facebook ads" in Facebook. In addition, our study shows that DSNs have a "smaller world" than GSNs. By taking the advantage of this fact, DSN services like Codebook may improve functions such as "Finding People and Artifacts". In addition, our comprehensive study of the community structure of DSNs showed that community structure exists and tends to stabilize over time. This knowledge enables researchers that use or examine DSNs to take advantage of these communities. This can be used, for example, for explicitly identifying "teams" in analysis of task resolution or for broadcasting information to a relevant subset of the community. Other potential research topics such as studying the effects brought about by key events including leadership changes within each community could be further explored based on our study as well.

ACKNOWLEDGMENT

Our thanks to Channy Yun and Mozilla developers for providing us useful insights to understand the Mozilla community structure.

REFERENCES

[1] H. Kwak, Y. Choi, Y. H. Eom, H. Jeong, and S. Moon, "Mining communities in networks: a solution for consistency and its evaluation," in Proceedings of the 9th ACM Internet Measurement Conference (IMC). New York, NY, USA: ACM, Nov. 2009, pp. 301-314.

[2] Y. R. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. L. Tseng, "Blog community discovery and evolution based on mutual awareness expansion," in WI '07: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence. Washington, DC, USA: IEEE Computer Society, 2007, pp. 48-56.

[3] M. E. J. Newman, "Modularity and community structure in networks," Proceedings of the National Academy of Sciences, vol. 103, no. 23, pp. 8577-8582, Jun. 2006.

[4] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: understanding microblogging usage and communities," in Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, ser. WebKDD/SNA-KDD '07. New York, NY, USA: ACM, 2007, pp. 56-65.

[5] Y. Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, "Analysis of topological characteristics of huge online social networking services," in WWW '07: Proceedings of the 16th international conference on World Wide Web. New York, NY, USA: ACM, 2007, pp. 835-844.

[6] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in Proceedings of the 19th international conference on World wide web, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 591-600.

[7] A. Nazir, S. Raza, and C. N. Chuah, "Unveiling Facebook: a measurement study of social network based applications," in IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement. New York, NY, USA: ACM, Oct. 2008, pp. 43-56.

[8] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," Aug. 2004.

[9] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," Physical Review E, vol. 69, no. 2, pp. 026 113+, Feb. 2004.

[10] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, ser. SIGSOFT '08/FSE-16. New York, NY, USA: ACM, 2008, pp. 24-35.

[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," Journal of Statistical Mechanics: Theory and Experiment, vol. 2008, no. 10, pp. P10 008+, Jul. 2008.

[12] J. Travers and S. Milgram, "An experimental study of the small world problem," Sociometry, vol. 32, no. 4, pp. 425-443, 1969.

[13] M. Marchiori and V. Latora, "Harmony in the small-world," Physica A: Statistical Mechanics and its Applications, vol. 285, pp. 539-546, 10/1, 2000.

[14] R. Guimerà, L. Danon, D. A. Guilera, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," Physical Review E, vol. 68, no. 6, pp. 065 103+, Dec. 2003.

[15] K. Crowston and J. Howison, "The social structure of free and open source software development," First Monday, vol. 10, no. 2, 2005.

[16] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in MSR '06: Proceedings of the 2006 international workshop on Mining software repositories. New York, NY, USA: ACM Press, 2006, pp. 137-143.

[17] S.-Y. Chan, P. Hui, and K. Xu, "Community detection of Time-Varying mobile social networks," in Complex Sciences, 2009, vol. 4, ch. 115, pp. 1154-1159.

[18] L. A. Adamic, B. A. Huberman;, A. L. Barab&aacute;si, R. Albert, H. Jeong, and G. Bianconi;, "Power-Law distribution of the world wide web," Science, vol. 287, no. 5461, pp. 2115a+, Mar. 2000.

[19] A. Begel, Y. P. Khoo, and T. Zimmermann, "Codebook: discovering and exploiting relationships in software repositories," in ICSE '10: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering. New York, NY, USA: ACM, 2010, pp. 125-134.

[20] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," Proceedings of the National Academy of Sciences, vol. 99, no. 12, pp. 7821-7826, Jun. 2002.

[21] A. MacCormack, J. Rusnak, and C. Y. Baldwin, "Exploring the structure of complex software designs: An empirical study of open source and proprietary code," MANAGEMENT SCIENCE, vol. 52, no. 7, pp. 1015-1030, Jul. 2006.

[22] J.Xu, Y. Gao, S. Christley, G. Madey. A Topological Analysis of the Open Souce Software Development Community, 2005.

[23] A. L. Barabasi, E. Bonabeau. Scale-Free Networks. Scientific American. Vol. 288, No. 5(2003), pp. 50-59

[24] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-Law distributions in empirical data," SIAM Review, vol. 51, no. 4, pp. 661-703, 2009.

[25] D. Surian, D. Lo, and E.-P. Lim, "Mining collaboration patterns from a large developer network." in WCRE'10, 2010