Network Structure of Social Coding in GitHub

Ferdian Thung, Tegawendé F. Bissyandé, David Lo, Lingxiao Jiang

▶ **To cite this version:**

Ferdian Thung, Tegawendé F. Bissyandé, David Lo, Lingxiao Jiang. Network Structure of Social Coding in GitHub. 17th European Conference on Software Maintenance and Reengineering (CSMR 2013), Mar 2013, Genova, Italy. pp.1-4. hal-00790772

HAL Id: hal-00790772
https://hal.science/hal-00790772v1

Submitted on 21 Feb 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Network Structure of Social Coding in GitHub

Ferdian Thung[1], Tegawendé F. Bissyandé[2], David Lo[1], and Lingxiao Jiang[1]
[1]*Singapore Management University, Singapore*
[2]*Laboratoire Bordelais de Recherche en Informatique, France*
{*ferdianthung,davidlo,lxjiang*}*@smu.edu.sg, bissyand@labri.fr*

***Abstract*—Social coding enables a different experience of software development as the activities and interests of one developer are easily advertized to other developers. Developers can thus track the activities relevant to various projects in one umbrella site. Such a major change in collaborative software development makes an investigation of networkings on social coding sites valuable. Furthermore, project hosting platforms promoting this development paradigm have been thriving, among which GitHub has arguably gained the most momentum.**

In this paper, we contribute to the body of knowledge on social coding by investigating the network structure of social coding in GitHub. We collect 100,000 projects and 30,000 developers from GitHub, construct developer-developer and project-project relationship graphs, and compute various characteristics of the graphs. We then identify influential developers and projects on this subnetwork of GitHub by using PageRank. Understanding how developers and projects are actually related to each other on a social coding site is the first step towards building tool supports to aid social programmers in performing their tasks more efficiently.

I. Introduction

Recently, developers have witnessed the emergence of platforms for social coding, such as GitHub[1] and Altassian BitBucket[2]. These platforms offer unique experiences to developers: they can broadcast their activities and/or listen to the activities of others; they can also investigate and leverage activities occurring in a variety of projects in one umbrella site.

The current momentum of social coding sites provides an opportunity for research on the impact of programmer networking in software projects. Recently, Dabbish *et al.* have investigated, through a series of interviews, the impact of transparency in GitHub [3]. Such studies are important as they help us to better understand the phenomenon of social coding. A good understanding of the characteristics of GitHub can indeed help researchers and practitioners to gain more of the insights that are needed to design better tools for supporting *social coders*. Furthermore, a thorough understanding of developer behaviors on GitHub will yield new ways for inciting more collaborations among developers.

In this study, we investigate GitHub, which is arguably the largest social coding site, containing more than 3 million repositories. We aim to extend the limited body of knowledge about social coding by constructing the network structure of projects and developers on GitHub and analyzing various characteristics of these networks.

The contributions of this work are as follows:

1) To the best of our knowledge, we are the first to investigate network structure of a social coding site (GitHub).
2) We highlight interesting network statistics of GitHub developers and projects.
3) We automatically locate influential projects and developers on GitHub by leveraging PageRank.

The remainder of this paper is structured as follows. In Section II, we present preliminary information on GitHub. In Section III, we introduce the various network statistics that we use as well as the PageRank algorithm. In Section IV, we present our research questions and their answers. We discuss related work in Section V. We conclude with future work in Section VI.

II. GitHub: A Social Coding Site

GitHub is a social coding site that uses Git[3] as its distributed revision control and source code management system. It implements a social network where developers are enabled to broadcast their activities to others who are interested and have subscribed to them. GitHub currently hosts over three million projects maintained by over one million registered developers. A given developer can participate in multiple projects and each project may have more than one developer. The GitHub social coding site is a developer-friendly environment integrating many functionalities, including wiki, issue tracking, and code review.

Within GitHub, there are pages for developers and pages for projects. An example of a GitHub page[4] related to the user *kemitche* (Keith Mitchell).This page includes information on kemitche's repositories (i.e., projects) and his recent public activities, such as committing code to a repository, opening an issue report, etc., which are seldom easily visible in other development environments. The page also shows several statistics that are often used on social networking sites, such as the number of other developers *following* him, the number of projects he is *watching*, etc. Such transparency is an interesting feature of GitHub and other social coding sites.

[1] http://github.com [2] https://bitbucket.org

[3] http://git-scm.com/ [4] https://github.com/kemitche

III. Methodology

In this section we describe our methodology for constructing a sample network from GitHub. We also introduce the statistics and the PageRank algorithm that we use for analyzing the network.

A. Network Construction

We construct two kinds of networks from GitHub data: a project-project network, and a developer-developer network. The project-project network is a graph of projects. This graph represents a network in which each node is a project, and where two nodes are connected if the corresponding projects have at least one common developer. We furthermore associate a weight to each edge of the graph; this weight corresponds to the number of developers that work together on both projects.

To construct this project-project network,a trivial solution is to check one project with every other project and look for the number of common developers. However this would be costly. To alleviate this computation issue, we perform the steps described in Algorithm 1. For each project, we first get the developers that work for it, we then find all the projects that the developers work for. This set of projects is typically of a small size. We then just compare the input project with all projects in the set.

Algorithm 1 Selecting Efficiently

```
Input: Projects // set of projects
Network ← ∅ // Project-project network
foreach project P_a in Projects do
    Developers ← listDevelopersInvolved(P_a)
    foreach developer D_a in Developers do
        smallSetProjects ← listProjects(D_a)
        foreach project P_b in smallSetProjects do
            link ← countCommonDevelopers(P_a, P_b)
            Network ← {Network, link}
return Network
```

In a developer-developer network, each node represents a given developer in our dataset. The corresponding graph contains an edge between two vertices when the corresponding developers work together in at least one common project. Similarly to the project-project graph, we associate a weight to each edge taking into account the number of projects where the two relevant developers work together. To build the developer-developer network, we proceed with the same methodology as for the project-project network.

B. Network Statistics

Various statistics can be computed to characterize a network. In this study, we primarily use a common metric, *node degree*, which, for a given node, considers the number of distinct nodes that are directly connected to it. We also rely on other common measurements, namely the *network diameter* and the *average shortest path*. The diameter of a network is the longest shortest path between all pairs of nodes in a network, while the average shortest path is the average of all shortest paths. We estimate the diameter and the average shortest path following the sampling approach as used by Surian *et al.* [9]. For our study, we sample 1000 nodes and compute the shortest paths for all possible combinations of these nodes.

C. PageRank

Introduced by Brin and Page, the PageRank algorithm for weighting web pages importance based on their links has gained popularity driven by its use in the Google search engine [2]. PageRank works in many iterations. In the initial iteration, the algorithm assigns the same PageRank score to all web pages. Then subsequent iterations update these scores: the score of a page p is distributed to the pages that p links to; each linked page receive $\frac{1}{|L_p|}$ of the score, where L_p is the set of pages that p links to. The PageRank score of a web page p at iteration i can thus be computed by the following equation:

$$PR(p,i) = \frac{1-r}{T} + r \sum_{q \in K_p} \frac{PR(q,i-1)}{|L_q|}$$

In this equation, r represents the probability that a web surfer would continue to surf (a.k.a. the damping factor), T is the number of web pages in the database of the search engine, K_p is the set of web pages that link to p, and L_q is the set of web pages that q links to.

IV. Empirical Evaluation

In this section, we describe our dataset and a set of research questions. Then, we present the results of our empirical evaluation to answer the research questions.

A. Dataset and Research Questions

Our dataset is comprised of 100,000 projects collected from GitHub using its API[5]. From these projects we have sampled 30,000 developers using GitHub's API. Using this dataset, we intend to answer the following research questions:

- RQ1 How strong are the relationships among projects?
- RQ2 How strong are the relationships among the developers?
- RQ3 Which projects are the most influential?
- RQ4 Which developers are the most influential?

B. Project-Project Relationship

To answer the first research question, we proceed in two steps: first, we compute the number of edges in the project-project graph. We have found 1,161,522 edges, meaning that 1,161,522 pairs of projects share at least one common developer. Second, we compute the degree of each node, i.e.,

[5] http://developer.github.com

the number of edges incident to this node in the project-project network. Figure 1 shows the degree distribution across the 100,000 projects of our dataset. We find that Figure 1, which shows a long tail, follows a power law distribution as is the case in most natural processes [8].

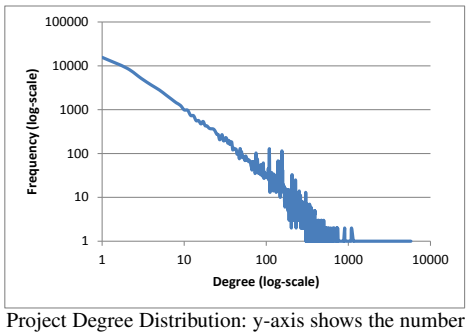

Figure 1. Project Degree Distribution: y-axis shows the number of projects having given edge degrees.

Finally, we measure the diameter of the largest connected component and the average shortest path between sampled project nodes. The diameter is 9 and the average shortest path, 3.7. These numbers are lower than the findings reported for many real networks [6], implying that project networks are actually more interconnected than human networks. Project networks, as defined in Section III, indeed, only require one common developer to establish a connection between two projects.

> Project networks are more interconnected than human networks. Software projects can always benefit from the expertise of different developers from different background.

C. Developer-developer Relationship

To answer the second research question, we proceed with the same steps as in the first question. The number of edges computed in the developer-developer network is 23,678,445, revealing that many pairs of developers share at least one common project. Note that this number is significantly larger than the number of edges in the project-project graph.

Figure 2 illustrates the degree distribution in the developer-developer network. This distribution does not form a long tail, as some projects involve an excessively large number of developers. Thus, each developer in such projects will share a connection with all other developers in the same project, resulting in both a high degree and a high frequency. Nonetheless, we still notice that, overall, some developers share a project with many other developers while the majority of developers share projects with a few developers.

The diameter of the largest connected component is 5 and the average shortest path is 2.47. We compare these values to findings in studies on two networks, namely Facebook and Sourceforge. In their study on the Sourceforge project hosting platform, Surian *et al.* have shown that the average shortest-path among project developers is 6.55 [9], following the popular assumption of "six-degree-of-separation" [11].

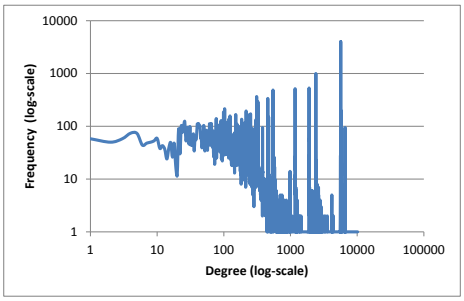

Figure 2. Developer Degree Distribution: y-axis corresponds to the number of developers having a given degree.

The average shortest path in Github is significantly lower, which suggests that the social coding concept actually enables more collaborations among developers. A recent study of the Facebook social graph has concluded that individuals on Facebook have potentially tremendous reach with an average shortest path of 4.7 [13]. The Github developer social network allows for even better reach as *developer-developer* relationships are less tight than *human-human* relationships in daily life social networks. Indeed, hundreds of developers may collaborate in a single project without even "knowing" each other.

> Social coding enables substantially more collaborations among developers.

D. Influential Projects

To identify influential projects, we run the PageRank algorithm described in Section III on the project-project network. Asides from its established effectiveness in measuring the importance of network nodes, as implemented in the Google search engine, PageRank is also known to be faster than many other importance score algorithms, including *Betweenness centrality* [4]. This property is indeed essential since the computation for thousands of nodes can be time consuming.

Project url	**PageRank**
https://github.com/mxcl/homebrew	0.0009862
https://github.com/rails/rails	0.0006378
https://github.com/lifo/docrails	0.0006370
https://github.com/joyent/node	0.0002161
https://github.com/rubinius/rubinius	0.0001678

Table I
TOP 5 MOST INFLUENTIAL PROJECTS

We detail in Table I the top-5 PageRank scores that the algorithm has produced after it was run for each project in the network. These influential projects provide libraries, programmer utilities and scripts and language implementations. The top-1 project is *homebrew* entitled "the missing package manager for OS X", which provides a package installer of UNIX tools for Mac users. This project has 7233 developers. It shares one or more developers with many other projects such as rails, docrails, homebrew-php, rvm, etc.

E. Influential Developers

To identify the influential developers, we run the PageRank algorithm in the developer-developer network. The

algorithm returns a score for every developer. The top-5 developers in terms of their scores are shown in Table II.

Developer		PageRank
Joshua Peek	josh[AT]joshpeek.com	0.00009536
Aman Gupta	aman[AT]tmm1.net	0.00008860
Steve Richert	steve.richert[AT]gmail.com	0.00008850
Michael Klishin	michaelklishin[AT]me.com	0.00008170
Josh Kalderimis	josh.kalderimis[AT]gmail.com	0.00008163

Table II
TOP 5 MOST INFLUENTIAL DEVELOPERS

The top-1 developer is Joshua Peek. This developer, who is part the core team of rails, the second influential project, works on 81 projects in collaboration with many others including Aman Gupta from the top-10 influential developers and others such as Sam Stephenson, Aaron Patterson, Mislav Marohnic, etc.

F. Discussion

From the above findings, we note that many projects share one or more developers with other projects. Social coding would benefit developers working on these projects as it provides a platform for developer activity to be transparent to others. Still we notice that many projects are not connected to one another and many developers do not share many projects. Thus there is a room to increase the number of collaborations in the network.

We also find some factors that cause developers to work together. Developers that work on common application types tend to work together since they have similar background. For example, the *rails* and *rubinius* projects which are strongly related to the adoption of Ruby share 88 common developers. Projects such as *node* and *symfony* that are in the list of influential projects have respectively 3820 and 1096 developers. These developers could work together on other projects once they get to know one another.

G. Threats to Validity

In this preliminary study, we only study a sample of projects and developers in GitHub. In our setting, analyzing the entire dataset is by far too expensive in terms of computing time, memory cost, and hard disk costs. This might cause a bias in the results. Nevertheless, we believe that this threat is limited as we consider a sizeable sample of 100,000 projects. In the future, we plan to mitigate this threat further by including more projects and developers.

Another threat to validity is that we consider two developers to be linked as long as they are involved in the same project. We do not differentiate when they work on the project, whether they work on the same code base, and whether they *actually* work together. Nonetheless, our findings offer initial interesting insights on social coding and its impacts.

V. RELATED WORK

There have been a number of studies that analyze network structure. Xu et al. and Surian et al. investigate the network structure of SourceForge [14], [9]. Hong et al. investigate network structure of developers involved in Mozilla Bug Tracking system [5]. Different from our study, both of these networks are not a social coding site. Other studies analyze the relationships between social media and software development. For example, Pagano and Maalej analyze the use of blogs in software development [7]. Bougie et al. and Tian et al. analyze the use of microblogs and Twitter in software development [1], [10]. Treude et al. investigate how developers ask and answer questions in StackOverflow [12].

VI. CONCLUSION AND FUTURE WORK

We have performed an empirical study on a a popular social coding site: GitHub. In this study, we extract information about 100,000 projects from GitHub and analyze both the project-project network and the developer-developer network. Our evaluation results show that distribution graphs in the project-project network generally follows a power law or long tail phenomenon while the developer-developer network generally does not. Nevertheless, we have shown that social coding indeed improves collaboration among developers: this conclusion can be inferred from the small value of the average shortest path in the largest community of developers. In the future, we wish to allocate more resources into extracting and computing statistics for a larger network of millions of developers contributing to a more extensive set of projects in GitHub.

REFERENCES

[1] G. Bougie, J. Starke, M.-A. Storey, and D. German, "Towards understanding twitter use in software engineering: Preliminary findings ongoing challenges and future questions," in *Web2SE*.

[2] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *WWW*, 1998.

[3] L. A. Dabbish, H. C. Stuart, J. Tsay, and J. D. Herbsleb, "Social coding in github: transparency and collaboration in an open software repository," in *CSCW*, 2012, pp. 1277–1286.

[4] L. C. Freeman, "A Set of Measures of Centrality Based on Betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, Mar. 1977.

[5] Q. Hong, S. Kim, S. C. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in *ICSM*, 2011, pp. 323–332.

[6] J. Leskovec and E. Horvitz, "Planetary-scale views on a large instant-messaging network," in *WWW*, 2008, pp. 915–924.

[7] D. Pagano and W. Maalej, "How do developers blog?: an exploratory study," in *MSR*, 2011, pp. 123–132.

[8] D. Sornette, *Critical Phenomena in Natural Sciences: Chaos, Fractals, Selforganization and Disorder: Concepts and Tools*, 2nd ed. Springer, 2004.

[9] D. Surian, D. Lo, and E.-P. Lim, "Mining collaboration patterns from a large developer network," in *WCRE*, 2010, pp. 269–273.

[10] Y. Tian, P. Achananuparp, I. N. Lubis, D. Lo, and E.-P. Lim, "What does software engineering community microblog about?" in *MSR*, 2012.

[11] J. Travers and S. Milgram, "An Experimental Study of the Small World Problem," *Sociometry*, vol. 32, no. 4, pp. 425–443, 1969.

[12] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?" in *ICSE (NIER)*, 2011.

[13] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph," *CoRR*, vol. abs/1111.4503, 2011.

[14] J. Xu, Y. Gao, S. Christley, and G. R. Madey, "A topological analysis of the open souce software development community," in *HICSS*, 2005.