Mining GitHub: A Literature Review

Weiliang Wang
Department of Computer Science
University of Victoria
Victoria, B.C. Canada
Email: wangweiliang61@gmail.com

***Abstract*—GitHub is the most well-known source code hosting site which serves as not only a collaborative coding platform but also a social media. Such a new open source environment has given rise to lots of research in different angles. This review of literature tries to summarize and reorganize all of these literature in a proper way, and to present both the current the situation and the future challenges.**

***Keywords*—GitHub, Collaborative Coding, Social Media, Data Mining, OSS**

I. Introduction

For the large and complex software projects, it is a challenge to well coordinate the collaboration between all the developers and to manage each developer's contribution. Based on this demand, Version Control System (VCS) was developed to record changes to a file or set of files over time so that you can recall specific versions later. [11] The traditional Centralized Version Control System (CVCS) requires developers to commit a change to a central repository, merge and resolve the conflicts. However, it has some severe downsides in that the entire history of the project lies in the central database. If the central database becomes corrupted, everything would be lost. [11][16]

Distributed Version Control System (DVCS) relaxes the requirement of CVCSs to have a central, master repository. Each developer owns a first-class repository of its own right, containing the entire project history.[16] DVCSs make developing, releasing and coordinating large open source software projects much more flexible than traditional CVCS. [30]

Among all these DVCSs, Git has gained the most momentum. It began its life in 2005 as a revision management system used for coordinating the Linux kernal's development. Over the years, Git has evolved by leaps and bounds due to its functionality, portability, effciency, and rich third-party adoption.[34]

Based on Git, some Web-based applications have been developed to host open source projects for free and to make it easy for project management using a Web-based user interface. It's convenient for developers to set up repositories, clone existing projects and commit their contributions.[34] Moreover, such applications lay much emphasis on the social aspect of software engineering.

Among all these social medias, GitHub is the most known one. GitHub users all have an account. They can not only broadcast their activities to others who are interested but also follow other users or projects they are interested in. Besides, users can explore some projects they might be interested in, share snippets and pastes with others, and write some blogs. By bringing these social networking features to DVCSs, it's of great help for communication and coordination.[14]

Generally speaking, GitHub is a social coding website, combining the social networking features and software engineering. Such a new environment gives rise to lots of research in different angles. Due to the social aspects introduced to GitHub, problems become more complicated. Sometimes both quantitative and qualitiative methods should be used for analysis. It could not be a good way to organize this literature review according to methodology, since the same problem could be analyzed using different methods. Therefore, the framework of this literature review is built on different research purposes.

The remainder of this paper is organized as follows: Section II summarized and reorganized different research topics around GitHub; Section III presented a discussion and gave the conclusion.

II. Literature Review of GitHub Mining

A. How Social Features Affect Software Engineering

The role of social media in software engineering is still not well understood, and yet the use of these mechanisms influences software development practices. [35] Thus there have been lots of research on it.

Paper [5] highlighted social media's impact for enabling new ways for software teams to form and work together. Potential is described for social media to both improve communication and coordination in software development teams and support the creation of new kinds of software development communities. Apart from that, some challenges are presented in terms of individual reputations and privacy.

Storey et al. (2010) [35] extracted several social media features such as Wikis, blogs, microblogs, tagging, feeds, social networking, meshups and crowdsourcing. In order to guide both future software development processes and tools, they proposed a set of pertinent research questions around community involvement, project coordination and management, as well as individual software development activities.

Tsay et al. (2012) [41] performed a quantitative analysis of project success of over 5,000 open source software projects hosted on GitHub. Based on two measures for software projects, Developer Attention and Work Contribution, they found little evidence that specific social media features in GitHub directly correlates with software project success, but they claimed that the observed effects could be mediated by the social features. However, the extent to which these findings are mediated remains to be studied.

In the paper [4], a semistructured interview was conducted with four leaders from four successful companies (GitHub, MSDN, Stack Exchange and TopCoder) to understand the role social networking plays in today's software development's world. In this interview, issues were explored such as the social software features used, incentives for encouraging participation, networks formation, trust building, challenges and future prediction, and so on. These four companies all provide a platform or a community where users can interact with each other in lots of ways. Such a social atmosphere greatly motivates the public intelligence and improves developing efficiency.

Choi et al. (2013) [13] presented both qualitative and quantitative analyses from which they derived some initial propositions regarding the impact of transparency on voluntary self-organization processes and decision mechanisms. Dabbish et al. (2012) [15] also held that the transparency is of great value for large-scale distributed collaborations and communities of practice, which can be supported by a series of interviews with some central and peripheral GitHub users. They found that people infer a lot from the activity information in GitHub. These inference can be used to better coordinate projects, improve their technical capability and improve personal reputation. Although transparency is so powerful, it's not a cure for all the ills. [14] Some developers don't feel comfortable when their activities can be completely seen by others, but the level of such discomfort is still not clear.

B. Tools for GitHub Mining

With GitHub gets more and more adoption, huge data of ultra-large-scale software repositories and developer activities has been flooding into it, which is such a treasure for the researchers. However, the prerequisite of mining these data is you already have it. Users can get access to a rich collection of data through an API provided by GitHub, but there still exist some challenges, mainly because the data is too huge and its scheme is not documented. [19] Therefore, in order to make it convenient enough for developers or researchers to acquire or analyze the data from GitHub, many extensive tools have been developed.

Gousios et al. (2012) [19] described GHTorrent project, which aims to offer a mirror of GitHubs data and event streams to the research community as a service in a scalable manner.

Dyer et al. (2013) [17] introduced Boa, a domain-specific language and infrastructure. It is to ease testing the hypotheses related with mining software repository (MSR). They also had implemented Boa and provided a web-based interface to Boas infrastructure. This infrustructure greatly reduced programming efforts and improved scalability.

Bissyande etc. (2013) [8] developed Orion, a unified platform for searching relevant software projects. Their prototype knowledge database was built with over 100,000 projects collected from the web not only from GitHub but also Google Code or Freecode that usually hosts some single-man projects. Orion facilitated the execution of various complicated queries that are not supported by traditional web search engines, but more data source should be added into their database.

C. How to Model GitHub

In order to better understand GitHub in a quantitative way, a mathematical model should be built. Among all the models adopted in the literatures, graph construction is the most wildly used to understand how developers and projects are connected.

In the chapter 7 of the book [31], Russell et al. (2013) provided an introduction to graph-oriented analysis using NetworkX based. knowledge was taught about how to take advantage of GitHubs rich data by constructing a graphical model of the data that can be used in a variety of ways. Similarly, Thung et al. (2013) [39] constructed developer-developer and project-project relationship graph using the data of 100,000 projects and 30,000 developers from GitHub. Several statistical metrics were calculated to characterize the graph, and PageRank Algorithm was used to find the most influential projects and developers. However, they just used a small proportion of developers and projects on GitHub, and more data should be collected. Besides, the way of graph constrcution also threaten the validity, because they think two developers can be connected as long as they share at least one project, and two projects can be connected as long as they share at least one developer. Although these limitations exist, graph analysis has been greatly adopted for different purposes.

Loyola et al. (2012) analysed GitHub in a different way. [23] To make it clear how developers interact with the projects being developed, they adopted Lotka-Volterra-based biological models based on the concept mutualism from social parasites, which is usually used to analyze host-parasite interaction. Some quatitative metrics were obtained, indicating that Open Source Software site like GitHub can be regarded as a mutualistic system, in the sense that both developers and repositories can co-exist and their populations converge to a steady state. However, more solid statistic analysis should be conducted to gain stronger confidence.

D. Mining Collaborative/Interactive Patterns

Since social media is playing a key role in GitHub, emphasis should also laid on developers' collaboration with other developers and their interaction with other projects. There have been a lot of research finding some interesting patterns.

1) Geographical Distribution: In order to see how widely and closely developers are interacted around the world, some research focuses on the geographical information using visualization techniques. Takhteyev et al. (2010) presented an empirical study of the geography of open source software development on Github. [37] Observations were found that developers are highly clustered and concentrated primarily in North America and Western and Northern Europe. Code contributions and attention presented a strong local bias. However, in paper [6], Xu et al. (2013) conducted a case study to find that developers from different regions are more likely to get connected. This conflict may be because developers have been interacted more widely during these years. Apart from that, Heller et al. (2011) applied visualization techniques to user profiles and repository metadata, providing a way for researchers to explore the question space and communicate initial insights. [20] Such patterns might be found including the effect of geographic distance on developer relationships, social connectivity and influence among cities, and variation in projectspecific contribution styles. Shrestha et al. (2013) used visualization to show the developers, developer locations and the frequency of commits based on the commit log in one integrated view. [32] But the user contribution was not considered, neither was which files were modified.

2) Diffusion Phenomena: In order to better understand project diffusion characteristics and the improvement of the popularity, Jiang et al. (2013) [21] collected 2,665 projects and 272,874 events, and analyzed topological characteristics and reciprocity of social graphs which was constructed based on 747,107 developers and 2,234,845 social links. Based on these data, they analyzed how project popularity is distributed across the GitHub social graphs, and characterized the role played by social links in project propagation. Albano et al. [1] analyzed the diffusion phenomena occur in GitHub via an innovative approach based on a notion of intrinsic time which can isolate the diffusion phenomenon from the evolution of the network.

3) Developer Assessment and Team Formation: Team formation is another typical problem, in which users are to find qualified partners that can satisfy the requirements of a collaborative task. An investigation was conducted on how such teams are formed on a social network based on millions of software repositories spanning a period of four years and hundreds of thousands of developers from GitHub. [24] They designed an algorithm that achieved significant improvement in finding effective teams as compared to naive strategies and scaled well with the size of the data.

Actually one usually finds his partners based on his impression for other users. In paper [26], Marlow et al. (2013) described a qualitative investigation of impression formation. They found that users usually form impressions around other users expertise based on history of activity across projects, and successful collaborations with key high status projects in the community. However, the impressions of expertise and ability studied in this paper was still not accurate enough, because the developers working on the same project might vary in technical skills.

Similarly in paper [33], by studying members of developer profile aggregators, an ecosystem was found that developers are assessing each other to evaluate whether other developers are interesting, worth following, or worth collaborating with. At the same time, developers are also self-conscious about being assessed, and thus polish their own profile. This mutual assessment is changing how software engineers collaborate and how they advance their skills.

Paper [46] examined the ability of the WikiTrust algorithm (content-driven reputation) to measure author quality in collaborative code development. They proved this technique was promising, just with a small number of false positives because of the differences between wikis and code repositories.

4) Developers' Behavior Patterns: In paper [36], some patterns were extracted from a large developer collaborations network at high and low level of details. Though it was from SourceForge.Net, it had its value as a reference for GitHub. A combination of graph mining and graph matching was employed to overcome the challenge of extracting subgraph from large graphs. Some observations were obtained: there are many collaboration clusters, and the number of nodes of the clusters and connectivity follow power law. Apart from that, the small-world phenomenon was observed where each developers in a connected network is separated on the average by approximately 6 hops. They also found that weak links resulting from transitive relationships between developers in the network are likely to evolve to direct collaborations.

Allaho et al. (2013) [2] studied the connectivity structure of the social networks of contributors and to investigate the effect of the different social tie structures on developers overall productivity to OSS projects. They also found that the social networks of OSS communities follow power-law degree distributions and exhibit small-world characteristics. The result demonstrated evident influence of the social ties on the developers overall productivity.

Paper [12] studied the herding phenomena on GitHub, in which the authors analyzed the participation patterns in 10 randomly sampled projects of Open Source Software Develoopement (OSSD) from GitHub. They found that 1) the patterns of contribution in OSSD projects with more developers who actively make uses of social features will exhibit clustering; 2) the signals generated by OSSD projects (e.g., extensive code commits, new release et al.) with greater importance are more likely to induce herding behavior from developers; 3) signals of events generated with higher frequency by OSSD projects are more likely to induce herding behavior from developers.

Tsay et al.(2012) [41]performed a quantitative analysis of project success of over 5,000 open source software projects hosted on GitHub. They used two measures, Developer Attention and Work Contribution, for projects' success, and fount that projects with a high level of developer multitasking tend to receive less Developer Attention, but greater Work Contribution. Success on both measures is strongly positively associated with greater concentration of work among a small number of developers.

Lee et al. (2013) [22] analyzed the activities of 544 GitHub developers working across 5,657 projects, with the purpose of examining how the network of developers and projects influence where developers choose to contribute. They obtained some very interesting phenomena of those "rockstars" (a group of extremely well connected developers), such that the rockstars' actions have a greater influence on their followers compared to regular developers, and the extent of such influence depends on the type of action, projects' age, and so on.

Authors of paper [7] investigated the actual adoption of issue trackers in software projects towards answering questions: 1) who cares about filing issues? 2) what kind of issues are reported in issue trackers? 3) what kind of correlation exist between issue reporting and the success of software projects? Based on an empirical study on about 20,000 projects, they made several observations: 1) issues are almost exclusively reported in large projects with big development teams led by developers having the most followers; 2) a large proportion of issue reporters are actually involved in the development of the project; 3) distributed development, recognized with project forking, contributes to an increase in the number of issue reporters; 4) there is only a small correlation between the numbers of issue reporters and the time-to-close delays.

In paper [27], qualitative, exploratory research was conducted with lead and core developers on three successful projects on GitHub to understand how OSS communities on GitHub measure success. Two findings were reported: 1) lead and core members of the projects we interviewed display a nuanced understanding of community participation in their assessment of success; 2) they attribute increased participation on their projects to the features and usability provided by GitHub.

GitHub users vary in coding skills, leading to many defects undiscovered, which can degrade software quality and is really a threat to the whole developing ecosystem on GitHub. Some researchers have been trying to solve this problem. Paper [29] conducted interviews with 33 active users of GitHub and an online questionnaire that was answered by 569 members of GitHub, trying to find how the social transparency on GitHub affects developers testing behaviors. They proposed several strategies for software developers and managers to positively influence the testing behavior in their projects. But this research has some limitations in terms of efficiency, effectiveness, and generalizability.

Some other researchers focused on the absence of an adequate testing suite. In paper [28], an approach was presented for reducing the effort required by project owners for extending their test suites. This approach was based on the phenomenon of drive-by commits, which indicates that valuable test cases and maintenance tasks would be completed by capable users, even though they might not be involved in that project.

5) Interaction Between GitHub and Q&A Websites: The booming of GitHub gave rise to occurance of lots of Q&A websites that are much related with GitHub. Lots of developers on GitHub also ask questions or post answers on those Q&A websites. Therefore, the role of Q&A websites, as well as the relationship between GitHub and Q&A websites is to be understood.

In order to understand what questions are being asked on such Q&A websites and better leverage this knowledge, Treude et al. (2011) [40] analyzed data from Stack Overflow, which is one of the best Q&A websites, to categorize the kinds of questions that are asked, and to explore which questions are answered well and which ones remain unanswered. The results showed that code reviews and conceptual questions are usually getting good answers.

Vasilescu et al. (2013) [42] laid their emphasis on the role of Q&A webistes in the work cycle of OSSD. They investigated the interplay between Stack-Overflow activities and the development process of projects on GitHub. Observations were made that active GitHub committers ask fewer questions and provide more answers than others. Moreover, active StackOverflow askers distribute their work in a less uniform way than developers that do not ask questions. Also, despite the interruptions incurred, the StackOverflow activity rate correlates with the code changing activity in GitHub.

E. Seeking Technical Experts on GitHub

Every developer has an account on GitHub, and has left so many activity traces when contributing to a project or interacting with others. All of these information provides a profile for each developer. Due to the transparency of such sites, judging developers' skills based on their online behavior is usually much more convincing than a resume written by themselves. Therefore, more and more companies had used social media sites for recruitment and candidate evaluation. The question comes to: how to make use of these information to find good developers?

TABLE I. SUMMARY OF EMPLOYER INFERENCES FROM PROFILE SIGNALS

Signal	**Inference**	**Signal reliability ease of verifiability**
1. Active open source involvement	Shared open source values	Reliable; Easy
2. Contributions accepted to high status project	Community acceptance of work, quality of contributions	Reliable; Hard
3. Project ownership	Soft skills: Initiative, project management	Reliable; Easy
4. Side projects	Passion for coding	Reliable; Hard
5. Number of watchers or forks of project	Project popularity	Unreliable; Easy

Capiluppi et al. (2013) [10] summarized the social media and respective signals that can be used to assess candidates' qualification, such like online experience, online reputation, and so on. However, there are still some problems. For example, some information might be fake, so recruiters should filter out the noise of excessive. Another challenge for employers is how to accept these indicators and how to formalize a unique, accepted template to summarize online experience. Overall, this paper just qualitatively acted as a guide or recommendation on how find candidates online, without any implementation.

In order to understand how profiles on the site are used to assess people during recruitment and hiring for software development positions, Marlow et al. (2013) [25] conducted interviews with some GitHub users. The results revealed that both employers and employees would use some specific signals as reference about the user's skills, and such signals can be summarized in Table 1.

Apart from that, some research is trying to find the technical experts in specific domains. Venkataramani et al. (2013) [44] proposed a model to rank the expertise of the developers on some Question and Answer (Q&A) websites. This model assigned each developer on GitHub some technical terms according to his profile or activities, which can reflect the domains this developer specialize in. Then some tags were extracted from the question on Q&A website. When mapping these tags to the developers' technical terms, some experts in the corresponding domain could be obtained. This model is still not good enough in the sense that the expertise of a developer is modelled only based on his proficiency in programming language concepts. Some more metrics should be added.

Driven by the fact that software developement nowadays relies much on third party libraries, Teyton et al. (2013) [38] came up with LIBTIC, an search engine that can find relevant library experts. This application is based on the assumption that a developer is a library expert if he used this library in his commitment.

F. *Mining Repositories*

As more and more outstanding developers commit to GitHub, a rich collection of good repositories has become such a treasure. The existing respositories could give developers valuable insights, and it would be a good strategy for reducing costs and streamlining software development processes to refer to public repositories of software. But it's still a challenge to find specific repositories that are wanted.

In [18], a tool was presented to support the analysis and selection of components and software applications that are of higher quality or best meet the quality requirements of a product. However, this tool was designed with limitation to only three technologies (java, python and php) and only 16 search terms.

In the paper [3] , based on 352 million lines of Java, authors built the first giga-token probabilistic language model of source code, which was significantly better at the code suggestion task than previous models. Some new complexity metrics were presented based on statistical analysis of large corpora.

Venkataramani et al. (2013) claimed that it is important to identify the domains to which the projects belong. [43] They proposed a model to cluster projects by mining latent semantics in a source code repository using the developer contribution network. These identified clusters are mapped to domains based on a taxonomy that was constructed using the metadata from a Q&A website.

Software projects rarely exist in isolation. Instead, a majority of projects is built on a network of people and ideas from many other projects. Using the GitHub APIs, Wagstrom et al. (2013) [45] presented a dataset of a large network of open source projects centered around Ruby on Rails, which provided insight into the relationships between Ruby on Rails and an ecosystem involving 1116 projects.

[9] Bissyande et al. (2013) take advantage of rich data on GitHub by conducting a large scale and comprehensive study that examine the popularity, interoperability, and impact of various programming language measured in different ways, such as lines of code, development teams, issues, etc.

III. Discussion and Conclusion

This review of literature nearly covered all the research topics around GitHub, presenting the current research scenario in this area. Researchers proposed lots of models to better represent GitHub structure, and adopted both qualitative and quatitative methods to analyze GitHub data. Various conclusions and patterns have been observed.

However, there also exist many challenges. For example, although graph analysis is so widely adopted and proves to work well for much analysis, the graph construction is still a problem, because different ways of construction may lead to different results.

Apart from all the covered topics, there are still some subarea waiting to be studied. Link prediction has been applied in some social networking websites but still remains to be analyzed on GitHub, which means how to predict the future links between developers and projects.

Acknowledgment

The author would like to thank Dr. Wu for giving so many instructions on how to write a good literature review. Besides, the author's thanks would also come to his supervisor Dr. German for providing much useful information.

References

[1] Alice Albano, Jean-Loup Guillaume, Sébastien Heymann, and Bénédicte Le Grand. A matter of time - intrinsic or extrinsic - for diffusion in evolving complex networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 202–206, New York, NY, USA, 2013. ACM.

[2] Mohammad Y. Allaho and Wang-Chien Lee. Analyzing the social ties and structure of contributors in open source software community. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 56–60, New York, NY, USA, 2013. ACM.

[3] Miltiadis Allamanis and Charles Sutton. Mining source code repositories at massive scale using language modeling. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 207–216, Piscataway, NJ, USA, 2013. IEEE Press.

[4] A. Begel, J. Bosch, and M.-A. Storey. Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder. *Software, IEEE*, 30(1):52–66, Jan 2013.

[5] Andrew Begel, Robert DeLine, and Thomas Zimmermann. Social media for software engineering. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, pages 33–38, New York, NY, USA, 2010. ACM.

[6] Xu Ben, Shen Beijun, and Yang Weicheng. Mining developer contribution in open source software using visualization techniques. In *Intelligent System Design and Engineering Applications (ISDEA), 2013 Third International Conference on*, pages 934–937, Jan 2013.

[7] Tegawende F. Bissyande, David Lo, Lingxiao Jiang, Laurent Reveillere, Jacques Klein, and Yves Le Traon. Got issues? who cares about it? a large scale investigation of issue trackers from github. In *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*, pages 188–197, 2013.

[8] T.F. Bissyande, F. Thung, D. Lo, Lingxiao Jiang, and L. Reveillere. Orion: A software project search engine with integrated diverse software artifacts. In *Engineering of Complex Computer Systems (ICECCS), 2013 18th International Conference on*, pages 242–245, July 2013.

[9] T.F. Bissyande, F. Thung, D. Lo, Lingxiao Jiang, and L. Reveillere. Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 303–312, July 2013.

[10] A. Capiluppi, A. Serebrenik, and L. Singer. Assessing technical candidates on the social web. *Software, IEEE*, 30(1):45–51, Jan 2013.

[11] Scott Chacon and Junio C Hamano. *Pro git*, volume 288. Springer, 2009.

[12] Joohee Choi, Junghong Choi, Jae Yun Moon, Jungpil Hahn, and Jinwoo Kim. Herding in open source software development: An exploratory study. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work Companion*, CSCW '13, pages 129–134, New York, NY, USA, 2013. ACM.

[13] Junghong Choi, Jinwoo Kim, Bruce Ferwerda, Jae Yun Moon, and Jungpil Hahn. Impact of social features implemented in open collaboration platforms on volunteer self-organization: Case study of open source software development. In *Proceedings of the 9th International Symposium on Open Collaboration*, WikiSym '13, pages 25:1–25:2, New York, NY, USA, 2013. ACM.

[14] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Leveraging transparency. *Software, IEEE*, 30(1):37–43, Jan 2013.

[15] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: Transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1277–1286, New York, NY, USA, 2012. ACM.

[16] B. de Alwis and J. Sillito. Why are software projects moving from centralized to decentralized version control systems? In *Cooperative and Human Aspects on Software Engineering, 2009. CHASE '09. ICSE Workshop on*, pages 36–39, May 2009.

[17] R. Dyer, Hoan Anh Nguyen, H. Rajan, and T.N. Nguyen. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 422–431, May 2013.

[18] G. Farah and D. Correal. Analysis of intercrossed open-source software repositories data in github. In *Computing Colombian Conference (8CCC), 2013 8th*, pages 1–6, 2013.

[19] G. Gousios and D. Spinellis. Ghtorrent: Github's data from a firehose. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 12–21, 2012.

[20] Brandon Heller, Eli Marschner, Evan Rosenfeld, and Jeffrey Heer. Visualizing collaboration and influence in the open-source software community. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, pages 223–226, New York, NY, USA, 2011. ACM.

[21] Jing Jiang, Li Zhang, and Lei Li. Understanding project dissemination on a social coding site. In *Reverse Engineering (WCRE), 2013 20th Working Conference on*, pages 132–141, 2013.

[22] Michael J. Lee, Bruce Ferwerda, Junghong Choi, Jungpil Hahn, Jae Yun Moon, and Jinwoo Kim. Github developers use rockstars to overcome overflow of news. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 133–138, New York, NY, USA, 2013. ACM.

[23] P. Loyola and In-Young Ko. Biological mutualistic models applied to study open source software development. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*, volume 1, pages 248–253, Dec 2012.

[24] Anirban Majumder, Samik Datta, and K.V.M. Naidu. Capacitated team formation problem on social networks. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1005–1013, New York, NY, USA, 2012. ACM.

[25] Jennifer Marlow and Laura Dabbish. Activity traces and signals in software developer recruitment and hiring. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 145–156, New York, NY, USA, 2013. ACM.

[26] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. Impression formation in online peer production: Activity traces and personal profiles in github. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 117–128, New York, NY, USA, 2013. ACM.

[27] Nora McDonald and Sean Goggins. Performance and participation in open source software on github. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 139–144, New York, NY, USA, 2013. ACM.

[28] R. Pham, L. Singer, and K. Schneider. Building test suites in social coding sites by leveraging drive-by commits. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 1209–1212, May 2013.

[29] Raphael Pham, Leif Singer, Olga Liskin, Fernando Figueira Filho, and Kurt Schneider. Creating a shared understanding of testing culture on a social coding site. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 112–121, Piscataway, NJ, USA, 2013. IEEE Press.

[30] P.C. Rigby, E.T. Barr, C. Bird, P. Devanbu, and D.M. German. What effect does distributed version control have on oss project organization? In *Release Engineering (RELENG), 2013 1st International Workshop on*, pages 29–32, May 2013.

[31] Matthew A. Russell. *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. O'Reilly Media, Inc., 2013.

[32] A. Shrestha, Ying Zhu, and B. Miller. Visualizing time and geography of open source software with storygraph. In *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on*, pages 1–4, Sept 2013.

[33] Leif Singer, Fernando Figueira Filho, Brendan Cleary, Christoph Treude, Margaret-Anne Storey, and Kurt Schneider. Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 103–116, New York, NY, USA, 2013. ACM.

[34] D. Spinellis. Git. *Software, IEEE*, 29(3):100–101, May 2012.

[35] Margaret-Anne Storey, Christoph Treude, Arie van Deursen, and Li-Te Cheng. The impact of social media on software engineering practices and tools. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, pages 359–364, New York, NY, USA, 2010. ACM.

[36] D. Surian, D. Lo, and Ee-Peng Lim. Mining collaboration patterns from a large developer network. In *Reverse Engineering (WCRE), 2010 17th Working Conference on*, pages 269–273, Oct 2010.

[37] Yuri Takhteyev and Andrew Hilts. Investigating the geography of open source software through github, 2010.

[38] C. Teyton, J.-R. Falleri, F. Morandat, and X. Blanc. Find your library experts. In *Reverse Engineering (WCRE), 2013 20th Working Conference on*, pages 202–211, Oct 2013.

[39] F. Thung, T.F. Bissyande, D. Lo, and Lingxiao Jiang. Network structure of social coding in github. In *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, pages 323–326, 2013.

[40] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 804–807, New York, NY, USA, 2011. ACM.

[41] Jason T. Tsay, Laura Dabbish, and James Herbsleb. Social media and success in open source projects. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion*, CSCW '12, pages 223–226, New York, NY, USA, 2012. ACM.

[42] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Stackoverflow and github: Associations between software development and crowdsourced knowledge. In *Social Computing (SocialCom), 2013 International Conference on*, pages 188–195, 2013.

[43] R. Venkataramani, A. Asadullah, V. Bhat, and B. Muddu. Latent co-development analysis based semantic search for large code repositories. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pages 372–375, Sept 2013.

[44] Rahul Venkataramani, Atul Gupta, Allahbaksh Asadullah, Basavaraju Muddu, and Vasudev Bhat. Discovery of technical expertise from open source code repositories. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 97–98, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[45] P. Wagstrom, C. Jergensen, and A. Sarma. A network of rails a graph dataset of ruby on rails and associated projects. In *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, pages 229–232, May 2013.

[46] Andrew G. West and Insup Lee. Towards content-driven reputation for collaborative code repositories. In *Proceedings of the Eighth Annual International Symposium on Wikis and Open Collaboration*, WikiSym '12, pages 13:1–13:4, New York, NY, USA, 2012. ACM.