Co-membership, Networks Ties, and OSS Success: An Investigation Controlling for Alternative Mechanisms of Knowledge Flow

Gang Peng
California State University Fullerton
gpeng@fullerton.edu

Feng Yu
Youngstown State University
fyu@ysu.edu

Yiqun Peng
QingDao University
pengyiqun@163.com

Abstract

Co-membership has been considered as a major mechanism for constructing social networks, but it has met many criticisms over time for failing to control for alternative mechanisms for knowledge flow. Although social networks constructed in online environment can reduce such possibilities, it is not without limitations. One possible mechanism for learning and knowledge flow is direct watching and observation. This study investigates the impact of co-membership taking into account the alternative mechanism of watching under the setting of OSS development at GitHub. It finds that both co-membership and watching contribute positively to OSS success, and thus shows the co-existence of both experiential learning and vicarious learning for OSS development. Moreover, it finds the impact of co-membership is much stronger than watching. While the impact of co-membership may be biased in prior literature, this study confirms that co-membership is indeed an effective mechanism for constructing online social networks for knowledge flow.

1. Introduction

Network ties are known to channel knowledge and expertise among social actors, and further influence their performance or decision-making [1]. Network ties can arise from various forms of mechanisms such as friendship, alliance, mobility, and advice, and one of the most important mechanisms is co-membership, where two entities are connected by a member belonging to both entities [2-4]. Members of an entity, e.g., team, project, firm, organization, can simultaneously participate in other entities, and therefore they become co-members of these entities, and they can potentially channel expertise and knowledge across the connected entities. The effect of co-members on knowledge diffusion and eventually performance has been documented in many studies under various settings. For example, business board-interlock can diffuse managerial practices and expertise across firms [5], co-members of TV production teams can bring in knowledge and expertise for movie production [4], and software development projects can benefit from knowledge leveraged by co-members [2]. As such co-membership has been used to construct social networks both online and offline. However, there has been some criticisms about the impact of co-membership on knowledge flow: while co-membership is important, there are other possible mechanisms through which learning and knowledge flow can occur [5], thus confounding the impact of co-membership. Therefore whether co-membership is indeed an effective mechanism for constructing social networks is called into question. Indeed, prior studies have shown inconsistent results, suggesting no or even negative relationship exists between co-membership and performance [6, 7]. Therefore, to establish a cleaner impact of co-membership, alternative mechanisms for learning and knowledge flow need to be controlled.

However, in real world, there are many alternative mechanisms that need to be controlled to establish a cleaner impact of co-membership. Obviously the simplest way to tease out their impact is to eliminate them altogether. In this regard, online or virtual environment presents an ideal setting to reduce the possibility of alternative mechanisms—individuals online are distributed worldwide and many of them might have never met before or will never meet in person, and as such the only mechanism for them to interact with each other is through online platforms.

However, even under the virtual environment, there are other possible mechanisms through which knowledge and expertise can flow. The most obvious and often cited mechanism is direct watching or lucking [8]. Prior studies have commented that individuals can directly watch other projects' development activities online and learn from them without joining these projects as members [9]. Thus

URI: http://hdl.handle.net/10125/50423
ISBN: 978-0-9981331-1-9

the following questions remain open: 1) Does co-membership really matter for performance and decision-making and thus can serve as an effective mechanism for constructing social networks? 2) As direct watching is often unobservable, how to control for it as an alternative mechanism for learning and knowledge flow? 3) Which mechanism, co-membership or watching, is more influential or effective for knowledge flow?

We intend to address the above questions in this study. Specifically we use GitHub to examine the impact of co-membership and watching for learning and knowledge flow.[1] GitHub is currently the most popular hosting website for open source software (OSS) project development. It possesses the features afforded by traditional platforms such as SourceForge.net and thus allows us to trace project co-membership. At the same time, it also exhibits certain features of social media such as watching, thus allowing us to tease out the learning effect due to direct watching [10]. By controlling for both co-membership and watching, we find that: 1) Co-membership indeed plays a critical role for learning and knowledge flow, even after controlling for watching. 2) Projects also learn from each other through the mechanism of watching. 3) The impact of co-membership is much stronger than watching.

2. Literature review

OSS development has gained increasing popularity in recent years. Many studies have been devoted to studying the success of OSS development. Among the various perspectives to OSS research, the one that has received increasing attention is that of social network theory, which argues that social actors are embedded in their relations, and they are connected by network ties, which can channel valuable information and facilitate learning and knowledge flow, and lead to improved outcomes and performance [1].

Social network ties can arise from many mechanisms. One of the frequently invoked mechanisms is co-membership. For example, prior literature suggests that member mobility can be effective for knowledge flow and performance improvement [11]. In OSS community, OSS project consists of multiple developers or project members, who in turn might participate in other projects. Here, a developer who belongs to two or more projects is called a "co-member" between the projects the developer concurrently participate in. Correspondingly, OSS projects that share one or more co-members are called connected projects. When a co-member works on a connected project, he can exchange ideas and discuss issues with other members through such tools as discussion forums, email lists, and tracker systems [12], and subsequently he learns from others and gains expertise and knowledge from participating in the project. Indeed, learning has been identified as one of the most significant motivations for OSS participation [13]. Furthermore, when the co-member works on the focal project, other members on the project learn from him as well. Therefore, through learning, knowledge and expertise can flow from the connected project to the focal project. Effectively, co-membership constitutes network ties between OSS projects and acts as conduits for knowledge flow across boundaries of OSS projects. Through co-members, useful information and knowledge, including innovative ideas and techniques for OSS development, can be channeled across the connected projects, influencing the performance of OSS projects, and more specifically the success of OSS projects [2].

In literature, project co-membership has been widely adopted to construct network ties among OSS or wiki projects. Table 1 shows some prior studies that have adopted co-membership to construct online social networks. A common finding of these studies is that co-membership is an effective mechanism for knowledge flow. We also examine if these studies have controlled for alternative mechanisms that can possibly lead to knowledge flow. However, we did not find evidence for controlling for alternative mechanisms such as watching.

Table 1: Sample Studies Using Co-membership

Studies	Levels of Analysis	Related Research Questions
[14]	Individual	How does collaboration network structure affect the contribution behavior in Wikipedia?
[15]	Individual	How does prior collaboration network affect developers' choice of newly initiated OSS projects to participate in?
[16]	Individual	How does participation in industry events relate to entrepreneurs' brokerage positions in informal industry networks?
[17]	Project	How does network position affect the market value of collaborative user-generated content?
[18]	Project	How does different types of network ties affect OSS project success?

[1] GitHub url: http://github.com

[19]	Project	How does social network structure affect the success of OSS projects?
[20]	Project	How does social capital affect knowledge flow in OSS development?
[21]	Project	How does network structure interact with human actions in affecting OSS success?
[22]	Project	How does network structure and network content affect technology adoption in OSS development?
[23]	Project	How does developers' attention to external projects may dampen OSS project success?
[24]	Project	How does social networks influence OSS project license choice?
[25]	Project	How does structural capital affect OSS project success?
[26]	Project	How does a project founder's network position affect time to release user-generated open source products?
[27]	Project	Are there knowledge spillovers in the network of open-source projects?
[28]	Project	How well does an affiliation network predict information quality on Wikipedia?

As commented by prior studies, the omission of controlling for alternative mechanisms is mainly due to the fact that there are virtually no way to track these mechanisms under traditional development platforms such as SourceForge, Wikipedia, or many other virtual environments [8]. However, more recent platforms like GitHub not only provide features afforded by SourceForge or Wikipedia, but also afford features of social media, so that developers and projects can directly watch others that are of interest to them [29].

GitHub not only provides a traceable project repository via Git, but it also acts as a social networking virtual space for individuals [30]. Just like other social applications, developers can "follow" other developers or "watch" other projects by subscribing them to a feed with frequent updates of their activities [9, 10, 29]. Figure 1 shows a snapshot of a typical project at GitHub.[2] The project has 5 members who have made 389 commits to its code repository. Most relevant to this study, it is being watched by 41 developers at the time. Usually, popular projects tend to be watched more and have more followers [31].

[2] It was taken on May 1, 2017.

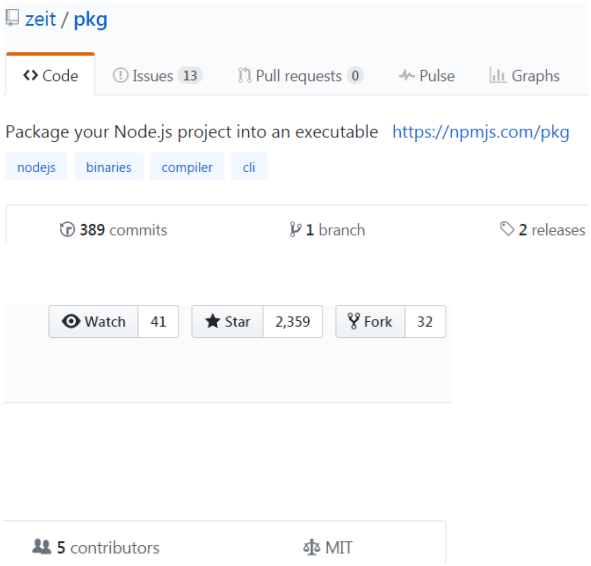

Figure 1. Snapshots of A GitHub Project

3. Theory and hypotheses

The literature on learning and its impact has a long history [32]. Learning can occur at various levels and through various mechanisms. Learning can be realized through ones' own experience or the experience of others [33]. Through learning, individuals, teams, and organizations accumulate stocks of knowledge, which can be applied to future activities. In this research, we adopt the view that team learning is an aggregate of individual learning as a result of actions and interactions among team members when they create, share, and integrate unique knowledge and information that can be applied in future situations [34-36].

3.1 Learning through co-membership

When members work on a project, they learn by working on the project, and when they move on to new projects or work concurrently on other projects, they apply what they learnt to the other projects [4]. Equivalently, these members become knowledge reservoirs and when they move, they carry the knowledge, expertise, and experience with them [11]. Learning from one's own experience has been referred to as experiential learning [37]. Experiential learning is particularly effective for gaining tacit knowledge that cannot be acquired easily through other types of learning. The impact of experiential learning on software development has also been observed. For example, when developing the first real-time online air ticket reservation system, *SABRE*,

many developers had participated in a prior project *SAGE*, and consequently, *SABRE* was able to not only benefit from technical innovations from *SAGE*, but also avoid many of the development pitfalls in system requirements, programming, and project management [38]. Similarly, the development of *FreeBSD*, an OSS Unix-like operating system, benefited greatly from *386BSD*, a relatively mature and stable operating system software, as many of the developers on *FreeBSD* used to work on *386BSD* [39, 40]. At project level, through co-membership, the focal project is connected to other projects which share these co-members. The more connected projects the focal project has, the more knowledge and expertise can potentially flow into the focal project, increasing the odd of its success:

H1: *OSS project success is positively associated with the number of connected projects*.

3.2 Learning through watching

In social computing workspace, developers can watch the activities of other projects. At GitHub, it has been observed that learning is one of the important motivations for observing other projects or users [9]. Once a project is set to be watched, all the activities of the project will be forwarded to the follower automatically through feeds [9, 10]. Therefore, through watching, the follower can examine and keep updated of the development activities of the watched project and learn from them.

Distinct from experiential learning or learning from one's own experience, learning from others' experience is referred to as vicarious learning [41, 42]. Vicarious learning is important for OSS development. First, OSS development consumes scarce resources, such as time, energy, cognitive, and computational efforts [43]. By taking advantage of others' experience and expertise, the focal project can economize the cost in decision-making, save their scarce cognitive efforts and resources, and improve the odds of making the right decision. Second, vicarious learning helps apply the experience and expertise of other projects to the focal project, and yields insights that potentially can increase the success of the focal project. Third, vicarious learning can also reduce risks associated with decision-making. Uncertainty is intrinsic in OSS project development [30]. When faced with the many tasks of software development, the focal project observes the actions of other projects and take into consideration the experience and lessons of others. In doing so, they can reduce the uncertainty associated with project development and enhance the success of the project.

At project level, members of a focal project may watch many other projects, and through watching, a form of vicarious learning, the focal project as a whole can gain knowledge and expertise, accumulate experience, and apply them to the focal project to enhance the odds of its success:

H2: *OSS project success is positively associated with the number of projects that the focal project is watching*.

The implication here is that, similar to co-membership, watching constitutes another mechanism for building online social networks. The activity of watching establishes the network ties between the focal project and the projects being watched. While ties based on co-membership is bidirectional, ties based on watching is unidirectional, representing knowledge flow from the projects being watched to the focal project.

3.3 Co-membership vs watching

As discussed above, co-membership and watching represent two different mechanisms for learning. A practical question for developers to ask is: which mechanism is more effective? In the context of OSS development, we believe co-membership is more effective than watching in leading to project success. First, being able to work on the connected projects directly allows developers to gain first-hand experience and knowledge, which affords them confidence in applying what they have learnt to the focal project. Although watching can speed up and economize the cost of learning, the fact that these experiences and knowledge are obtained second-hand can potentially cast doubt on their applicability to the focal project. Second, experience and knowledge though watching lack details and accuracy, thus they are hard to implement for the focal project. Third, compared to second-hand information, direct experience through co-membership lasts longer in memory and can be recalled and acted upon easier when needed [44]. Therefore, although both co-membership and watching are expected to be effective channels for learning, the former tends to be more powerful since the first-hand learning tends to be more relevant, and last longer:

H3: *The impact of co-membership is stronger than that of watching in affecting OSS project success*.

4. Datasets, variables, and method

The datasets we use for this study are from GitHub. Since first established in February 2008, GitHub has grown rapidly into the worlds' leading hosting website for OSS development. GitHub integrates a number of social features which allows users and their activities to be visible within and across OSS projects [9]. At GitHub, project members are those who make code contribution to a project.

We took two snapshots of the whole projects at GitHub, one on January 8, 2016 and the other one on November 11, 2016. Two datasets are built from them. The first one is used to construct independent variables used in this study, and the second one, together with the first one, are used to construct the dependent variable.

There are 25,364,494 projects in the first dataset. However, as noted by other researchers, many of the projects are inactive [45]. Therefore, we restrict our sample to projects that have made any commits during the study period, and this reduces the sample to 1,417,028. We further restrict the sample to projects that are not forked from any other projects, and the sample is further reduced to 1,158,021.[3] Since many of the projects are for individual use other than programming, we further restrict the sample to those having more than one member, and this leaves 308,127 projects which are used to construct the social networks as describe later. All the network metrics in this study are based on the project universe of these projects.[4] Figure 2 shows the counts of the projects using different programming languages.

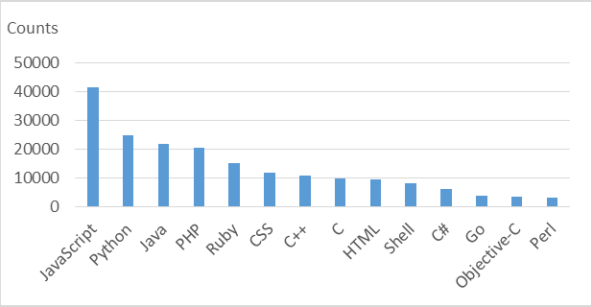

Figure 2. Project Counts by Programming Languages at GitHub

As can be seen from Figure 2 that Java is one of the major programming languages used at GitHub, and therefore, we focus on projects that use Java.[5] There are 21,786 Java projects among them.

We construct the independent variables using the first dataset. The independent variables include alters, watched alters, project size, project age, and project experience. Detailed variable definitions are provided in Table 2.

Table 2. Variable Definitions

Variables	Definitions
Log(commits)	The logarithm of commits made by a focal project between Jan and Nov 2016. It measures the *OSS project success*.
Alters	The total number of projects connected by the co-members between the focal project and its connected projects, divided by 10. It measures the *impact of co-membership*.
Alters watched	The total number of projects that are watched by the focal project, divided by 10. It measures the *impact of direct watching*.
Project size	The total number of contributors to a focal project
Project age	The age (in months) of a focal project in month since its registration with GitHub till Jan 2016
Project experience	The average experience (in months) of a focal project members since they registered with GitHub till Jan 2016

From the first dataset, we also construct two social networks using the co-membership and the watching mechanism respectively. The construction of the social network through co-membership is as follows: First, each member of a focal project is identified; second, for each member, the connected projects are identified; third, all unique connected projects are counted for the focal project as the number of alters through co-membership.

The watching mechanism to constructing social network follows the same way: First, each member of a focal project is identified; second, for each member, projects that are watched by that member are identified; third, all unique projects that are watched by the focal project are counted as alters watched.

We also present the kernel density estimation of variable *alter* and *alters_watched* in Figure 3 and 4. They show that both variables are heavily right skewed. To reduce the impact of outliers, we further restrict our sample to projects that have less than 50 alters and 100 watched alters.

[3] As commented by prior studies, forking makes it difficult to clearly identify the commits by the projects that are being forked [29, 42, 44].

[4] It needs to point out that working on such a large number of projects is not easy, and to speed up the processing speed, we made use of Amazon AWS big data platform.

[5] Although we restrict the analysis to Java projects, the network variables are derived from the whole project universe, as discussed earlier.

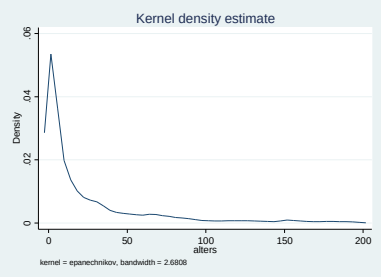

Figure 3. Kernel Density Estimate of Alters

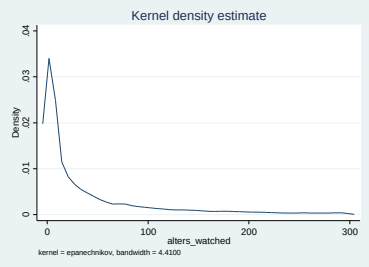

Figure 4. Kernel Density Estimate of Alters_watched

For the dependent variable, OSS project success, we use the logarithm of the number of commits made by the focal project during the study period, i.e., from January to December 2016—this represents the incremental changes made in the commits during the time period [46]. Through the time lag, we use independent variables to explain the dependent variable that are affected by them. Ordinary least squares (OLS) model is used for estimation.

5. Results

We first present the descriptive statistics and correlation matrix in Table 3. It shows that log(commits) is positively associated with both alters and watched alters, and this is consistent with our hypotheses.

Table 3. Descriptive Statistics and Correlation Matrix

Variables	1	2	3	4	5
Log (commits)	___				
Alters	0.097**	___			
Alters watched	0.087**	0.395**	___		
Project size	0.237**	0.300**	0.250**	___	
Project age	–0.019*	0.228**	0.198**	0.175**	___
Project experience	–0.014	0.414**	0.366**	0.040**	0.476**

Notes: N=14,626; *p<0.05; **p<0.01.

The OLS estimation results are shown in Table 4. Model 1 is with only the control variables, and establishes the baseline results. The coefficient on project size is positive and significant and that on project age is negative and significant (p<0.01). These suggest that as projects attract more contributors, more commits are made, which is as expected. On the other hand, as time passes, projects tend to make less commits. This echoes the observation that most of commits are made in the early stage when projects first get started; however, the coding frenzy may subside as projects mature.

Table 4. Estimation of OSS Project Success

Independent Variables	Model 1	Model 2	Model 3	Model 4
Alters		0.069** (0.014)	0.057** (0.014)	0.278** (0.040)
$Alters^2$				−0.059** (0.010)
Alters_watched			0.027** (0.007)	0.061** (0.0020)
$Alters\_watched^2$				−0.005 (0.003)
Project size	0.149** (0.005)	0.141** (0.005)	0.137** (0.005)	0.133** (0.005)
Project age	−0.007** (0.001)	−0.008** (0.001)	−0.007** (0.001)	−0.008** (0.001)
Project experience	−0.001 (0.001)	−0.001 (0.001)	−0.002* (0.001)	−0.004** (0.001)
R^2	0.060	0.062	0.063	0.065

Notes: N=14,626; dependent variable is log(commits); estimated coefficients and their associated standard errors (in parentheses) are listed under each model. *p<0.05; **p<0.01.

Model 2 adds *alters*, and Model 3 adds *alter*s and *alters_watched*. While the coefficients on *alters* are both positive and significant in both models, the coefficient in Model 3 is lower than that in Model 2. The implication is that without controlling for alternative mechanism of watching, and impact of co-membership will be biased upward, thus echoing the caveat in prior literature that it is necessary to control for alternative mechanisms when estimating the impact of co-membership [22].

Model 4 further adds variables $alters^2$ and $alters\_watched^2$, the quadratic terms of *alters* and *alters_watched*. The coefficient on $alters^2$ is negative and significant. This suggests that excessive number of alters may backfire, hurting the development of the focal project. This makes sense—co-membership among OSS projects requires developers to contribute codes to the participated projects, and as the number of alters continues to grow, developers will need to allocate and divert more of their limited resources across multiple projects, eventually hurting

the development of the focal project. On the other hand, although the coefficient on *alters_watched*^{2} is negative but is insignificant—watching other projects obviously requires less effort from the developers and thus incurs less cost to the focal project.

Taking into account the results from the four models, it can be seen that the coefficients on *alters* and *alters watched* are both positive and significant (p<0.01), thus H1 and H2 are supported. Furthermore, the coefficient on *alters* is much higher than that on *alters_watched*, in either Model 3 or Model 4 (p<0.01), therefore H3 is also supported.

6. Discussion and conclusion

Co-membership has been proposed as an effective mechanism for learning and knowledge flow. However, prior results have been hampered by the lack of controlling for alternative mechanisms [6, 7]. Even though online environment provides an ideal setting for studying the efficacy of co-membership, the possibility of alternative mechanisms still exists. For example, watching is very common for OSS development. In this study, we take advantage of the social computing platform afforded by GitHub to address three research questions we proposed earlier: 1) Does co-membership really matter for performance and decision-making? 2) How to control for direct watching as an alternative mechanism for knowledge flow? 3) Which mechanism is more influential or effective for knowledge flow? Since GitHub allows project members to record their watching behavior, leaving trace of watching, and thus we can effectively control for watching in this study. Consequently, we identify two different mechanisms for learning in this study: experiential learning through co-membership and vicarious learning through watching. We empirically show that both of these two mechanisms, or two forms of learning, are effective for knowledge flow; however, co-membership is more effective than watching. Our study makes several theoretical and practical contributions.

Although prior studies have shown the impact of co-membership, they do not control for the alternative mechanism of watching. Therefore, results from prior studies tend to be biased. In this study, after controlling for alternative mechanism, we show that learning and knowledge flow in the form of co-membership is indeed supported, even after controlling for alternative mechanisms. With the accumulation of first-hand experience, developers learn and accumulate knowledge from connected projects and apply it to the focal project to improve efficiency and economize cost. When faced with the constraints of time, cost, and most importantly uncertainty of the project development, project members are motivated to take advantage of learning from their own experience and apply what they have learnt to the focal project.

Literature suggests that vicarious learning is an effective way to gain access to valuable knowledge and information [8, 41, 42]. In OSS development, the social computing platforms such as GitHub also provide opportunities for vicariously learning through watching. We find that OSS development exhibits strong characteristics of vicarious learning through watching—observing peer projects which had accumulated relevant expertise and knowledge affords the focal project the opportunity to learn the second-hand information and knowledge from their peers.

With the presence of both experiential learning and vicarious learning, one important question to ask is which mode of learning is more effective. The answer to this question has significant implication for individuals and organizations as well. Our study reveals that in the context of OSS development, experiential learning has stronger impact than vicarious learning. However, we caution that quite often developers are constrained not only by the time and resources they possess, but also by their limited access to social networks to gain the needed information, thus they do not always have the luxury to decide on which mode to pursue. When time and resources allowing, developers may well explore the problems at hand by themselves through experiential learning; otherwise developers are probably better off to take advantage of the knowledge and experience of others through vicarious learning.

Lastly, although we show that the impact of co-membership tends to be biased without controlling for alternative mechanisms, co-membership remains to be an effective mechanism for learning and knowledge flow in online settings such as OSS development. The implication is two-fold: it shows that prior studies on OSS development and virtual communities based on co-membership is indeed valid and effective; at the same time, it points out the necessity for controlling for watching, an alternative mechanism for constructing online social networks, for future studies. Given the burgeoning number of studies on online social networks at GitHub [29, 31, 47-49], this seems particularly important.

There are several future research directions. In this study, due to the huge number projects, constructing the social networks is very time consuming and computational intensive. Therefore, we only use the Java projects to test our hypotheses. One of the future directions is to use projects of other

languages such as JavaScript or Python to validate the results of this study. Second, we use project commits to measure the success of OSS projects. However, there are other possible metrics to be considered, such as project quality and complexity, which we plan to explore in the future. Third, the effectiveness of the two mechanisms of learning obviously depends on the individuals who establish these two mechanisms, and therefore controlling for the skill or experience of the co-members or watchers represents another direction of our future research.

Acknowledgement:
This work was partially supported by AWS Programs for Research and Education (formerly Amazon Research and Education Grant).

References

[1] R. S. Burt, *Structural Holes: The Social Structure of Competition*, Oxford: Oxford University Press, 1992.

[2] R. Grewal, G. L. Lilien, and G. Mallapragada, "Location, location, location: How network embeddedness affects project success in open source systems," *Management Science*, vol. 52, no. 7, pp. 1043-1056, Jul, 2006.

[3] N. H. Lamb, and P. Roundy, "The `ties that bind` board interlocks research: A systematic review," *Management Research Review*, vol. 39, no. 11, pp. 1516-1542, 2016.

[4] A. Zaheer, and G. Soda, "Network evolution: The origins of structural holes," *Administrative Science Quarterly*, vol. 54, no. 1, pp. 1-31, Mar, 2009.

[5] P. R. Haunschild, and C. M. Beckman, "When do interlocks matter?: Alternate sources of information and interlock influence," *Administrative Science Quarterly*, vol. 43, no. 4, pp. 815-844, Dec, 1998.

[6] E. M. Fich, and L. J. White, "Why do CEOs reciprocally sit on each other's boards?," *Journal of Corporate Finance*, vol. 11, no. 1-2, pp. 175-195, Mar, 2005.

[7] N. Fligstein, and P. Brantley, "Bank control, owner control, or organizational dynamics: Who controls the large modern corporation?," *American Journal of Sociology*, vol. 98, no. 2, pp. 280-307, Sep, 1992.

[8] S. Rafaeli, G. Ravid, and V. Soroka, "De-lurking in virtual communities: a social communication network approach to measuring the effects of social and cultural capital," in Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.

[9] L. Dabbish *et al.*, "Social coding in GitHub: Transparency and collaboration in an open software repository.." pp. 1277-1286.

[10] A. Begel, J. Bosch, and M.-A. Storey, "Social networking meets software development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder," *IEEE Software*, vol. 30, no. 1, pp. 52-66, 01/01/2013, 2013.

[11] L. Argote *et al.*, "Knowledge transfer in organizations: Learning from the experience of others," *Organizational Behavior and Human Decision Processes*, vol. 82, no. 1, pp. 1-8, May, 2000.

[12] J. Xu, S. Christley, and G. Madey, "Application of social network analysis to the study of open source software," *Economics of Open Source Software Development*, J. Bitzer and P. Schroder, eds.: Elsevier Science, 2006.

[13] Y. L. Fang, and D. Neufeld, "Understanding sustained participation in open source software projects," *Journal of Management Information Systems*, vol. 25, no. 4, pp. 9-50, Spr, 2009.

[14] X. Q. Zhang, and C. Wang, "Network positions and contributions to online public goods: The case of Chinese wikipedia," *Journal of Management Information Systems*, vol. 29, no. 2, pp. 11-40, Fal, 2012.

[15] J. Hahn, J. Y. Moon, and C. Zhang, "Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties," *Information Systems Research*, vol. 19, no. 3, pp. 369-391, Sep, 2008.

[16] W. Stam, "Industry event participation and network brokerage among entrepreneurial ventures," *Journal of Management Studies*, vol. 47, no. 4, pp. 625-653, Jun, 2010.

[17] S. Ransbotham, G. C. Kane, and N. H. Lurie, "Network characteristics and the value of collaborative user-generated content," *Marketing Science*, vol. 31, no. 3, pp. 387-405, May-Jun, 2012.

[18] G. Peng, Y. Wan, and P. Woodlock, "Network ties and the success of open source software development," *Journal of Strategic Information Systems*, vol. 22, no. 4, pp. 269-281, Dec, 2013.

[19] O. Temizkan, and R. L. Kumar, "Exploitation and exploration networks in open source software development: An artifact-level analysis," *Journal of Management Information Systems*, vol. 32, no. 1, pp. 116-150, 2015.

[20] R. Mendez-Duron, and C. E. Garcia, "Returns from social capital in open source software networks," *Journal of Evolutionary Economics*, vol. 19, no. 2, pp. 277-295, Apr, 2009.

[21] J. Wang, M. Y. Hu, and M. Shanker, "Human agency, social networks, and FOSS project success," *Journal of Business Research*, vol. 65, no. 7, pp. 977-984, Jul, 2012.

[22] G. Peng, and D. Dey, "A dynamic view of the impact of network structure on technology adoption: The case of OSS development," *Information Systems Research*, vol. 24, no. 4, pp. 1087-1099, Dec, 2013.

[23] S. Daniel, and K. Stewart, "Open source project success: Resource access, flow, and integration,"

Journal of Strategic Information Systems, vol. 25, no. 3, pp. 159-176, Oct, 2016.
[24] P. V. Singh, and C. Phelps, "Networks, social influence, and the choice among competing innovations: Insights from open source software licenses," *Information Systems Research*, vol. 24, no. 3, pp. 539-560, Sep, 2013.
[25] P. V. Singh, Y. Tan, and V. Mookerjee, "Network effects: The influence of structural capital on open source project success," *MIS Quarterly*, vol. 35, no. 4, pp. 813-829, Dec, 2011.
[26] G. Mallapragada, R. Grewal, and G. Lilien, "User-generated open source products: Founder's social capital and time to product release," *Marketing Science*, vol. 31, no. 3, pp. 474-492, May-Jun, 2012.
[27] C. Fershtman, and N. Gandal, "Direct and indirect knowledge spillovers: The ``social network`` of open-source projects," *Rand Journal of Economics*, vol. 42, no. 1, pp. 70-91, Spr, 2011.
[28] G. C. Kane, and S. Ransbotham, "Content and collaboration: An affiliation network approach to information quality in online peer production communities," *Information Systems Research*, vol. 27, no. 2, pp. 424-439, Jun, 2016.
[29] Y. Yu *et al.*, "Exploring the patterns of social behavior in GitHub." pp. 31-36.
[30] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in GitHub." pp. 356-366.
[31] A. Lima, L. Rossi, and M. Musolesi, "Coding together at scale: GitHub as a collaborative social network," 2014/07/09, 2014.
[32] D. A. Levinthal, and J. G. March, "The myopia of learning," *Strategic Management Journal*, vol. 14, pp. 95-112, Win, 1993.
[33] L. Argote, and E. Fahrenkopf, "Knowledge transfer in organizations: The roles of members, tasks, tools, and networks," *Organizational Behavior and Human Decision Processes*, vol. 136, pp. 146-159, Sep, 2016.
[34] V. U. Druskat, and D. C. Kayes, "Learning versus performance in short-term project teams," *Small Group Research*, vol. 31, no. 3, pp. 328-353, Jun, 2000.
[35] A. C. Edmondson, "The local and variegated nature of learning in organizations: A group-level perspective," *Organization Science*, vol. 13, no. 2, pp. 128-146, Mar-Apr, 2002.
[36] A. P. J. Ellis *et al.*, "Team learning: Collectively connecting the dots," *Journal of Applied Psychology*, vol. 88, no. 5, pp. 821-835, Oct, 2003.
[37] G. P. Huber, "Organizational learning: The contributing processes and the literatures," *Organization Science*, vol. 2, no. 1, pp. 88-115, Feb, 1991.
[38] W. Scacchi, "Managing software engineering projects: A social analysis," *IEEE Transactions on Software Engineering*, vol. 10, no. 1, pp. 49-59, 1984.
[39] T. T. Dinh-Trong, and J. M. Bieman, "The FreeBSD project: A replication case study of open source development," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 481-494, Jun, 2005.
[40] bjk. "*A Brief History of FreeBSD*," April 19, 2017.
[41] P. Ingram, and J. A. C. Baum, "Opportunity and constraint: Organizations' learning from the operating and competitive experience of industries," *Strategic Management Journal*, vol. 18, pp. 75-98, Sum, 1997.
[42] A. Terlaak, and Y. Gong, "Vicarious learning and inferential accuracy in adoption processes," *Academy of Management Review*, vol. 33, no. 4, pp. 846-868, Oct, 2008.
[43] C. Casalnuovo *et al.*, "Developer onboarding in GitHub: The role of prior social links and language experience." pp. 817-828.
[44] J. P. Bonardi, and G. D. Keim, "Corporate political strategies for widely salient issues," *Academy of Management Review*, vol. 30, no. 3, pp. 555-576, Jul, 2005.
[45] E. Kalliamvakou *et al.*, "The promises and perils of mining GitHub." pp. 92-101.
[46] Y. Yoshikawa, T. Iwata, and H. Sawada, "Collaboration on Social Media: Analyzing Successful Projects on Social Coding."
[47] B. Heller *et al.*, "Visualizing collaboration and influence in the open-source software community." pp. 223-226.
[48] N. Kerzazi, and I. El Asri, "Knowledge flows within open source software projects: A social network perspective," *Advances in Ubiquitous Networking 2*, R. El-Azouzi *et al.*, eds., Singapore: Springer, 2017.
[49] F. Thung *et al.*, "Network structure of social coding in GitHub." pp. 323-326.