Predicting Software Revision Outcomes on GitHub Using Structural Holes Theory

Abstract

Many software repositories are hosted publicly online via social platforms. Online users contribute to the software projects not only by providing feedback and suggestions, but also by submitting revisions to improve the software quality. This study takes a close look at revisions and examines the impact of social media networks on the revision outcome. A novel approach with a mix of different research methods (e.g., ego-centric social network analysis, structural holes theory and survival analysis) is used to build a comprehensible model to predict the revision outcome. The predictive performance is validated using real life datasets obtained from GitHub, the social coding website, which contains 32,962 pull requests to submit revisions, 20,399 distinctive software project repositories, and a social network of 234,322 users. Good predictive performance has been achieved with an average AUC of 0.84. The results suggest that a repository host's position in the ego network plays an important role in determining the duration before a revision is accepted. Specifically, hosts that are positioned in between densely connected social groups are likely to respond more quickly to accept the revisions. The study demonstrates that online social networks are vital to software development and advances the understanding of collaborations in software development research. The proposed method can be applied to support decision making in software development to forecast revision duration. The result also has several implications for managing project collaboration using social media.

Keywords: software development, social network analysis, structural holes theory, survival analysis, predictive modeling

1 Introduction

Managing software projects is a difficult task. Software needs to be constantly maintained to support companies in a dynamic business world. Revising computer software in the development process and in later stages is unavoidable. Firms revise software for different purposes, e.g., to increase productivity, to meet customer demands, and to comply with regulations. Tracking changes of software versions (often known as software version control) allows companies to continuously support the development of software projects [1-3]. While companies benefit from computer software, they often find managing software projects challenging, due to budget limit, skill shortage, and other factors. Projects in IT often overrun in cost and schedule and yet fail to deliver the expected benefits [4]. As a result, software revision becomes a necessary but difficult task for companies to succeed in IT projects. Companies can improve project performance in a number of ways, such as by utilizing internal/external talents and nourishing better managerial practices with rigorous quality checks [4].

Open source software (OSS) communities attempt to tackle some of these issues using the wisdom of the crowd by embedding social features in software development. Increasingly, more software projects are hosted on public, open websites like SourceForge (sourceforge.net) and GitHub (github.com). Open source software merits the attention of many different stakeholders. End users might use OSS as an alternatives to commercial solutions, for example, open source operating systems, office tools, and image processing. Organizations, realizing the shortcomings of closed source commercial software [5], are interested in investigating the impact of open source software and adopting managerial practices to achieve better business outcomes [3, 6]. Among the many benefits, one significant advantage of hosting software projects online is the ability to allow online users to interact with each other, so that OSS websites become a social media platform for users to form a community.

Researchers are seeking to understand the impact of social networks on organizations [7, 8]. The social network, which links different users, has been proven to be an instrumental piece of organizational performance [9, 10] and brings both opportunities and challenges to organizations [11-16]. Therefore, networks from the social media are relevant to study software development and project management issues in organizations. In OSS communities, active users contribute by submitting revisions to software repositories. Once submitted, maintainers of the software repository choose whether to accept or reject the revision. In general, maintainers accept revisions that will improve the quality of the software, and reject revisions that are unsatisfactory or untrustworthy.

The acceptance of the revision is an essential topic in the context of OSS. Past studies have sought to develop explanatory and predictive models to study acceptance [17-20]. It is important to know the time it takes for a revision to become accepted or rejected, as the timing is an important issue in project delays and the software life cycle [21-23]. However, none of these studies explored the link between revision outcome and the structure of the social media networks. Yet studying this link is relevant as companies are now adopting the practices of the OSS community to improve their performance in software development [3, 24].

The goal of this paper is to investigate the predictive power of social network data on OSS revision outcome and the time elapsed before a revision is accepted. Capitalizing on findings within OSS, this paper will provide practical solutions to support organizations in managing software projects by taking into account the social network context. This paper uses social networks and structural holes theory together with survival analysis. Social network data demonstrates the complex networks of connections among users. The structural holes theory offers solid theoretical background for the structural aspects of the social networks, and links the social networks to individual outcomes in software development. The survival model utilizes the operationalized social network information, and provides a comprehensible statistical model to predict the revision outcome over time. Empirical data collected from the GitHub event archive involves 20,399 software projects repositories, 234,322 users, and 32,962 revisions. Statistical methods are used to validate the prediction results on the datasets, such as the Area Under the Curve (AUC) [25, 26] and cross validation [27, 28]. The literature review follows in section two. The third section explains the research method, the data collection procedure and the evaluation metrics. Lastly, section four is a discussion of the research results.

2 Literature review

2.1 The “social” development of Open Source Software

The impact of open source software has been discussed for more than 10 years [29, 30]. Stakeholders in the domain have gradually recognized the identifying traits of OSS. The research area covers but is not limited to transparency [31], trustworthiness [32], team organization [21, 33, 34], and performance measures [17]. The influence of the OSS tunnels through the boundary between the open and the closed source industries. Open source software is used in closed source contexts [29], and closed source software might become open [35]. Moreover, software companies are trying to adopt some of the good practices from the OSS community [3, 6, 24, 36]. With the emerging role of social media and social software websites, many OSS projects are not only hosted online, but also in a “social” context. Online repository hosting websites such as GitHub enable social features, allowing users to interact with each other and contribute to OSS. People are able to follow each other, join different organizations and subscribe to different repositories. The OSS projects, alongside the contributors and their social networks, become publically available. Three main streams of research focus on the social network features [37, 38], the pull request acceptance [18-20], and using data generated from OSS for forecasting [17, 30].

Social coding is a unique experience for users to develop software projects interactively. Research is invested in issues raised from “coding socially.” A revision submitted to a software project might not always be the best revision [19]. Often a revision (sometimes also called a “pull request” on websites like GitHub) will experience delays before being accepted or rejected by the repository administrators, who are responsible for reviewing the revision and managing the repositories [20, 39]. Some research has attempted to investigate this problem and identified potential predictors of revision outcome, such as project and individual characteristics [18, 19, 33]. Researchers identified useful features to determine the acceptance of the revision requests, such as whether a contributor included test code before submitting the request [39], or worked closely with the repository administrator (sometimes called integrator) resulting a number of comments and feedbacks [20]. Previous successful experience of submitting revisions will help future acceptance [19]. However, given the complex nature of software development, some of the findings contradict others. For instance, while intensive communication between the users helps to strengthen social relations, it also may signal that the revision might be problematic, that more conversation is needed to resolve the problems. As a result, the role of the comments as a proxy for communication in predicting the acceptance could be both positive and negative [20, 40]. Similarly, while including a testing unit in the revision might help with testing the code, many repository administrators choose to evaluate the revision without the submitted tested code because they might have their own way of testing [39]. Studies attempting to characterize the revision requests with technical indicators are often limited by sample size and programming language types, and, so far, have failed to deliver a more general picture of the collaboration process. Social coding is not just a technical/engineering process, but also a human decision process and a networking experience. While the social media platform allows everyone to contribute, it falls upon repository administrators to evaluate those submissions and to decide whether to trust contributors. The decisions by repository administrators to accept the revisions will have a direct impact on their projects, their co-workers, their communities, and potentially millions of the repository users worldwide. Managing the repository is more than a technical task; repository administrators often find it difficult to justify the acceptance of revision requests using only the technical indicators. Repository administrators often have to refer to other co-workers' reviews of the submitted contributions, and the connections to their co-workers to help them to make decisions in managing contributions [39]. In light of those findings, this study takes a relational perspective, using the available network data on the social media platform to predict revision outcomes.

2.2 Social network analysis

Social network analysis is an essential element in social science research [41-43]. Social network analysis has emerged due to the advances in information technology and promotes a better understanding of different research topics in engineering, business, economics and social sciences [9, 42, 44-46].

Ego-centric network analysis is a sub-branch in social network analysis that focuses on a set of particular nodes. In a typical social network, individual users are considered as nodes and the relationships (such as being friends, relatives, and colleagues) are the ties linking the nodes. In an ego-centric network design, research is based on one central (sometimes known as "focal") node, called the "ego," while the nodes linking the ego are called "alters." The ego network is a subgraph of the complete full social network. Different ego networks form small-world networks [41] and are sometimes called "social circles" [47]. The ego-centric network allows researchers to explore the social networks among a number of interested nodes with their directly connected "neighbors" even without having access to the full network structure. Gathering full network information is often difficult due to the size of the network and its ability to change over time. The ego-centric network is a widely used alternative to study social networks without exploring the whole network [41].

Sociologists have found that people's positions in the social networks are closely related to individual outcomes [48-51]. The connections in the social networks are an essential asset for people to gain access to vital information and resources to compete, to negotiate, and to innovate [52]. Because people have limited energy to maintain a finite number of relationships, the question remains of how to maintain the ties efficiently and effectively in order to benefit from a network of finite size. Burt discussed some of these issues in his structural holes theory [52, 53], arguing that social groups that are connected tend to share similar information and resources. People connecting between many different social groups tend to have a richer share of information and resources than those who establish connections only within a limited range of social groups. Structural holes occur when an ego node connects to all its neighbor nodes, while those neighbor nodes are not themselves connected [52, 54]. Structural holes are like a “buffer” between different disconnected social groups [52]. Structural holes are considered to be a structural advantage in social networks in order to access the valuable resources exchanged between unconnected social groups. Nodes with structural holes are expected to get faster promotion, and have more power to negotiate in the market [52]. Burt’s work introduces the idea of quantifying the positional information of individuals in social networks by measuring the distribution of the ties to different social groups. This allows the study of important individual outcomes, such as the time to accept revisions in OSS.

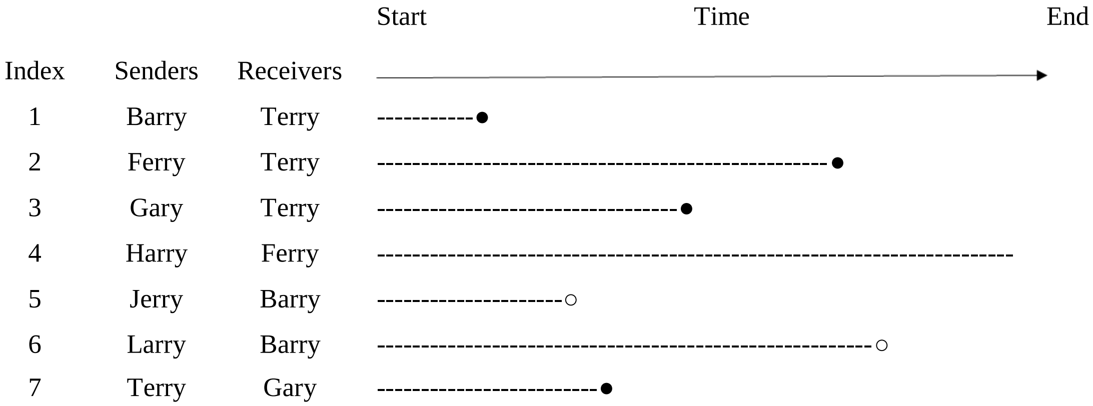
The previously conducted research naturally raises a set of new questions. Repository administrators often find it difficult to act upon a received revision, due to their limited availability of various resources, e.g., time, energy, knowledge and support from co-workers [39]. Nodes with structural holes have the potential to access more resources. Can structural holes measures predict revision outcome? Are the network of contributors and the network of repository administrators both equally predictive? How do social network data such as the structural holes measures generate quantitative, accountable, yet comprehensible insights for organizations to support their business decisions and to foster better managerial practices? This paper attempts to answer these questions using social network analysis.

2.3 Survival analysis

Analysis of the revisions should not only consider the status (accept/reject) but also the time it takes for the revision to be accepted. Survival analysis addresses the time to event problems. In a classical “survival setting,” the dependent variables are the status and time associated with the status. Typically in clinical trial studies, the dependent variables are whether the patients die or survive (status) and time [55-57]. Survival analysis can also be used to study the reliability of material in engineering, such as when exactly a type of material composite fails, as well as to study customer behavior and product lifetime in business [58, 59]. Survival analysis is different from other regression methods because it is designed exclusively to study the time to event behavior in the dataset. Survival analysis considers the status (such as death, recovery, failure, maturity, and so on) of the subjects and the time elapsed before having the status (time until death, time it takes to recover, and so on). Furthermore, as the data is gathered over time in a follow-up period, observations might not experience the event before the end of the follow up time. Alternately, observations may drop out. These observations are considered to be “censored.” The survival analysis is designed exclusively to make use of the censored data for the model inference. As a result, the survival model provides coefficient estimates that indicate the weights of the predictors, known as the hazard ratio [55, 57]. The hazard ratio is interpreted as the rate/speed to experience the event; hence, it is informative for research to discover the predictors that lead to faster or slower revision acceptance.

3 Methods

To study the time elapsed before a revision is accepted or rejected, one needs to consider the status and the time, respectively. In figure 1, there are a total of 7 revisions. Revision 1 was the first to be accepted, while revisions 7, 3, and 2 were accepted later. Revision 5 was rejected between the acceptance of 1 and 7. Revision 6 was eventually rejected, and revision 4 was censored during the follow-up period. Research methods associate these outcomes to the social networks of users, and identify the important variables that lead to acceptance/rejection of the revision.



Index	Senders	Receivers	
1	Barry	Terry	-----------●
2	Ferry	Terry	---●
3	Gary	Terry	----------------------------------●
4	Harry	Ferry	--
5	Jerry	Barry	---------------------○
6	Larry	Barry	--○
7	Terry	Gary	-------------------------●

Fig. 1. Time to event, the dependent variables
● Revision accepted
○ Revision rejected

3.1 Data collection

GitHub is the largest online software project platform, with 3.4 million users. At the end of 2014, GitHub hosted more than 2 million active repositories, according to statistics on GitHut (http://githut.info/).

GitHub includes many features of social media platforms. Users can follow one another, similar to other social media websites such as Twitter. In addition to their own repositories, GitHub users can also subscribe to other repositories in which they are interested, and become affiliated with different organizations. Users can also socialize by submitting revisions to others' repositories. A revision is formally known as a "pull request" on GitHub. In a pull request, a user revises a repository by adding or changing the code, and then sends it to the repository owner for approval.

This research project collected all pull requests sent from one user to another. Different pull requests generated from September 1st to September 7th, 2015 are collected. Pull requests are followed up for one additional week, from the date they are collected, shown in Table 1. A summary of the collected data is shown below in Table 1.

Table 1
A summary of the collected data

Dataset index	Follow up period (2015)	Issued pull requests	Accepted pull requests	Users	Repositories	Organizations
T1	09.01-09.07	8712	6142	11,229	5811	4176
T2	09.02-09.08	6466	3663	8785	5101	2975
T3	09.03-09.09	4714	2802	6839	3751	2054
T4	09.04-09.10	5818	1543	7374	4624	1179
T5	09.05-09.11	2359	1394	3464	2009	1281
T6	09.06-09.12	1761	1066	2643	1545	702
T7	09.07-09.13	3132	1772	4511	2575	1270
Total	\	32,962	18,382	39,694	20,399	20,234
Total number of users in the network: 234,322						

3.2 Social network analysis

The relations in a social network consisting of N nodes are presented in an adjacency matrix, which is a N-by-N matrix that represents all sets of possible relations concerning nodes. An ego-centric network has been extracted among the users, resulting in a dataset with 234,322 users in total. The users have a list of followers, and also follow a number of other users. The "following" relationship allows the construction of an unweighted directed social network, where $w_{ij} = 1$ means that node i follows node j .

Three variables extracted from the ego network are the effective size, the efficiency, and the hierarchy. All three variables are based on the structural holes theory, which argues that people should distribute their ties to distinctive social groups rather than limit their ties to one social group. Using Figure 2 as an example, there is an ego network of 7 people with Barry as the ego and the rest as alters. Barry has relationships with

Ferry, Gary and Harry while the others do not: Ferry, Gary, and Harry are isolated from the others by Barry. Hypothetically, if Ferry, Gary, and Harry have important information, Barry will be the next person to know, as he is their unique contact. These unique ties of Barry with Ferry, Gary, and Harry are considered to be non-redundant. Barry also connects to Jerry, Larry, and Terry. Those ties do not isolate Jerry, Larry, and Terry from each other. Ideally, one tie with either Jerry, Larry or Terry is enough for Barry to get information from all three, as those three are connected. If Barry connects to Larry, the other ties spent with Jerry and Terry can be considered "redundant" since these ties have no added value. In other words, the ego Barry establishes a connection to one social group including Jerry, Larry, and Terry, at a cost of maintaining three ties. The structural holes act as the separations between non-redundant contacts, for example, between Gary and Harry.

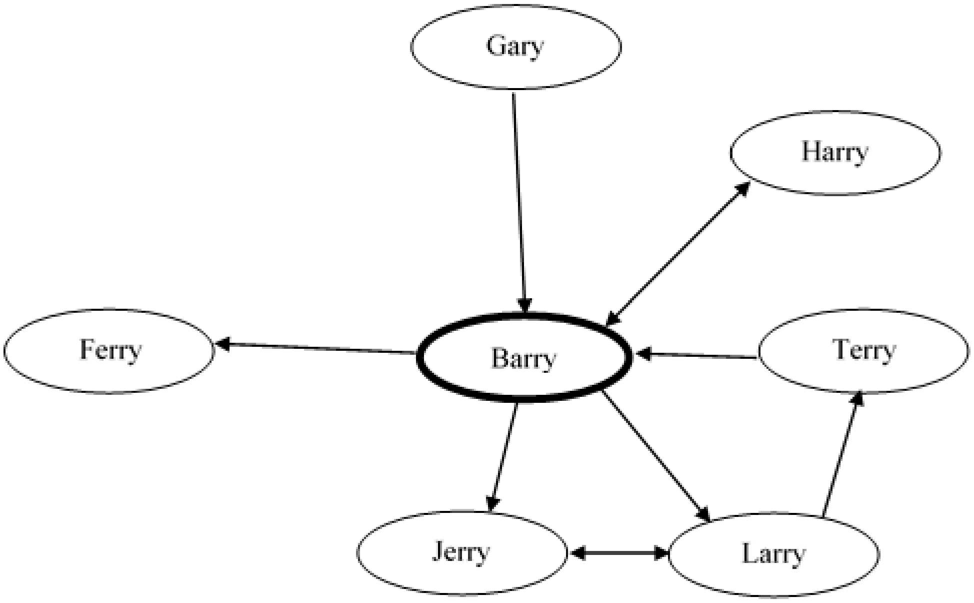


Fig. 2. Redundant and non-redundant ties in the ego network

Effective size (ES for short) computes non-redundant ties in the network for each node i.

$$ES_i = \sum_j \left[1 - \sum_q p_{iq} m_{jq}\right], q \neq i, j \tag{1}$$

where p_{iq} is the fraction of ties connecting i with node q:

$$p_{iq} = \frac{w_{iq} + w_{qi}}{\sum_j (w_{ij} + w_{ji})}, i \neq j \tag{2}$$

and m_{jq} is the relative strength of the tie between node j and q over the maximum tie strength of j:

$$m_{jq} = \frac{w_{jq} + w_{qj}}{\max\,(w_{jk} + w_{kj})}, j \neq k \tag{3}$$

For example, the effective size of node Barry (as node i) is the sum of the non-redundant ties to all other nodes (Larry, Terry…). A specific non-redundant tie in (1), such as the tie between Barry (node i) and Larry (node j) is computed based on the connectedness of Barry and Larry to other nodes such as Jerry (as node q). The effective size of Barry is higher when there are more disconnections among other nodes observed. The fewer connections in Barry's neighboring nodes, the more necessary it is that neighboring nodes need to refer to Barry, and the more information and resources Barry gets.

Effective size divided by total number of ties N_i (degree of node i) leads to efficiency (Ef for short):

$$Ef_i = \frac{ES_i}{N_i} \tag{4}$$

High efficiency means a high percentage of ties are non-redundant.

Constraints compute the direct and indirect ties between a pair of nodes that lead to the absence of structural holes:

$$c_{ij} = (p_{ij} + \sum_q p_{iq}\, p_{qj})^2\,, q \neq i, j \tag{5}$$

Taking constraints between Barry (node i) and Larry (node j) as an example, the more connections Larry has to other nodes (such as Jerry and Terry, indexed as node q), the more constraints Larry poses to Barry. This is because Larry will share the resources Jerry and Terry have with Barry.

Total constraints (TC) give an overview of the constraints of a node:

$$TC(i) = \sum_j c_{ij}\,, i \neq j \tag{6}$$

Hierarchy (H) is the extent to which a node's constraints (e.g., node i) concentrate on its relations to other nodes.

$$H_i = \frac{\sum_j \frac{c_{ij}}{TC/N_i} \ln\,(\frac{c_{ij}}{TC/N_i})}{N_i \ln\,(N_i)} \tag{7}$$

Effective size, efficiency, and hierarchy represent the social network information and will be used in the survival model as independent variables. Those structural holes variables measure how "important" a node is in the social networks. The overview of the variables will be given in the data collection section.

For clarity we note that in the context of social media networks, there can be many types of social ties [60]. For example, social media users might be connected because they directly communicate with each other, or – as is the case in this paper: they may just be linked because they follow each other.

3.3 Survival analysis

Assuming there are $\mathbb{N}$ observations with p variables (also known as covariates) in the dataset X, and each observation (a pull request) is one row in X, then $X_i = (x_{i1}, x_{i2}, x_{i3} \dots\ x_{ip})$ describes the characteristics of observation i. An observation contains information in a pull request: user who sends the pull request (sender), who receives it (receiver), their networks, repositories, organizations, and so on. There are $\psi = (1, 2, 3 \dots k)$ different timestamps at which the revisions are accepted.

Denote $h(t)$ as a hazard function to quantify the probability (Pr) that a revision will be accepted at given time t. $h(t)$ can be viewed as the limit within the time interval between t and $t + \Delta t$, which is the instantaneous potential of the revision being accepted at t. The probability is conditioned on $T \geq t$ meaning it only applies to those revisions that have not been accepted at t.

$$h(t) = \lim_{\Delta t \to 0} \frac{\Pr\ (t \leq T < t + \Delta t | T \geq t)}{\Delta t} \tag{8}$$

The most widely used survival model is the Cox proportional hazard model (Coxph). This semi-parametric model is robust and able to approximate other parametric models very well [55, 57]. The Cox proportional hazard model is used in this paper:

$$h(t) = \ h_0(t)e^{\beta X_i} \tag{9}$$

β are the model coefficients, and $h_0(t)$ is a baseline hazard function [61]. The baseline hazard is a function corresponds to the hazard for an individual having 0 for all the covariates, thus serving as the "baseline" comparing to other observations with non-zero value variables [55].

The inference of the Coxph model is based on the partial likelihood function $L(\beta)$:

$$L(\beta) = \prod_{\psi=1}^{k} \frac{e^{\beta X_{t(\psi)}}}{\sum_{m \in R_\psi} e^{\beta X_m}} \tag{10}$$

$L(\beta)$ is the joint probability from the distinctive timestamps $\psi = (1, 2, 3 \ldots k)$. At each specific time $t(\psi)$, a fraction is calculated. The $X_{t(\psi)}$ in the numerator in (10) encodes the variables of the revision accepted at the exact moment of the time $t(\psi)$. The denominator is the sum of a set of the remaining revisions that are not accepted up to time $t(\psi)$, known as the risk set R_ψ. Maximum likelihood estimation is performed to infer the parameters β [55, 57].

The Coxph model gives hazard ratios e^β to its variables. For instance, suppose a hazard ratio value of 1.3 is obtained for a particular variable to study the time to accept the revisions. Then an increase of 1 unit in the variable (while other variable values remain the same) leads to 1.3 times faster acceptance of the revisions, based on the model estimates, whereas an increase of 2 units leads to $1.3^2 = 1.69$ times faster acceptance. The Cox proportional hazard model has two major assumptions: non-informative censoring and proportional hazard. The non-informative censoring assumption was initially discussed in clinical studies, and concerns the fact that the patients drop out when they feel they are dying (e.g., when studying HIV) as time elapses [62, 63]. In this case, the censoring (dropout) time distribution might depend on the event time distribution, which might introduce errors in model estimation. Different from the clinical settings, this study does not suffer from such dropout syndrome, since it studies one user sending revisions to another. Hence, "self-withdraw" cannot occur. The non-informative censoring assumption will hold, as the major resource of the violation has been eliminated due to the uniqueness of the study and research design, which differs from the clinical studies. The proportional hazard assumption states that the effect of variables increases or decreases proportionally based on the hazard ratio. This assumption will be evaluated with the widely used weighted residual test [64] using the model results in section 4.1.

3.4 Dataset compilation

The datasets were constructed based on the pull requests, information about the senders who initiated the revisions, the targeted repositories, and the receivers (usually the owners or the administrators) who make decisions to accept revisions. The GitHub "who-follows-who" network is used as the social network data.

The collected network data is used to extract measures such as the total number of followers (often known as the in degree and the out degree in social networks), the effective size, the efficiency, and the hierarchy for the senders and the receivers. Users' affiliations to organizations, their own repositories, and their subscriptions to other repositories provide additional insights. Hence, the number of organizations, repositories, and subscriptions are incorporated as variables for each sender and receiver in the different revisions. Furthermore, the sender and receiver in a pull request may share organizations, repositories, subscriptions, and followers; these are known as the shared attributes and are coded with the prefix "share" in Table 2. The senders' and receivers' number of years ("Sy" and "Ry", respectively) on GitHub quantifies their experience and usage on the site. Furthermore, the variable "Is contributor" is used to identify whether the sender was previously listed as a major contributor of the project. An overview of all 23 variables is included in Table 2.

Table 2
List of variables
"Shared" measures the number of objects shared by a sender and a receiver in a specific revision. For instance "shareSub" is the number of subscriptions both a sender and a receiver have in common.

Variable names	Abbreviations	Data types	Max	Min	Mean	Std
Number of followers a sender has	sfer	Numeric	30	0	8.35	10.44
Number of followings a sender has	sfol	Numeric	30	0	6.19	9.51
Number of organization a sender is in	sorg	Numeric	30	0	0.68	1.54
Number of repositories a sender has	srep	Numeric	30	0	15.28	11.32
Number of subscriptions a sender has	ssub	Numeric	30	0	18.28	11.34
Number of followers a receiver has	rfer	Numeric	30	0	10.27	12.12
Number of followings a receiver has	rfol	Numeric	30	0	5.7	9.58
Number of organizations a receiver is in	rorg	Numeric	30	0	0.75	1.76
Number of repositories a receiver has	rrep	Numeric	30	0	17.25	11.4
Number of subscriptions a receiver has	rsub	Numeric	30	0	15.23	12.94
Number of common organizations shared	shareOrg	Numeric	5	0	0.08	0.32
Number of common repositories shared	shareRepo	Numeric	1	0	0	0.01
Number of common subscriptions shared	shareSub	Numeric	30	0	0.81	2.65
Number of followers and followings shared	shareF	Numeric	35	0	0.95	2.6
Effective size of the sender	sEffective	Continuous	59.36	0	11.59	14.92
Efficiency of the sender	sEfficiency	Continuous	1	0	0.57	0.35
Hierarchy of the sender	sHierarchy	Continuous	1	0	0.15	0.17
Effective size of the receiver	rEffective	Continuous	59.59	0	13.18	16.49
Efficiency of the receiver	rEfficiency	Continuous	1	0	0.51	0.39
Hierarchy of the receiver	rHierarchy	Continuous	1	0	0.12	0.15
Survival time	T (in hours)	Continuous	7	0	3.13	3.02
Acceptance	Status	Categorical	1	0	0.53	0.5
Is contributor	Contri	Categorical	1	0	0.7	0.46
Years of the sender registered	Sy (in years)	Continuous	7.68	0.02	2.8	1.96
Years of the receiver registered	Ry (in years)	Continuous	7.68	-0.18	3.05	1.95

3.5 Testing and evaluation

The research result is expected to be generalizable and reproducible; however, even high quality research may suffer from random, unavoidable errors. As a consequence, many research findings cannot be reproduced [65]. Thus, testing must be conducted in order to validate the research model. In data analytics, the statistical inference over the dataset is often influenced by the noise in the data. Cross validation is a method often suggested to deal with such situation [27, 28].

In this case, the dataset is split into two parts: one for building the statistical model, and the other for evaluating the model performance. In this way, the experiment reduces the potential bias when building and testing using a single dataset. A model tested with several different datasets is often considered more convincing if findings are consistent over different datasets [66].

The survival model is built based on the accepted revisions, and is then used to predict those that have not yet received the decision. For example, the pull requests' acceptance results on the dataset of T1 are used to build the survival model, which is then used to predict the future acceptance of pull requests on the other days, such as T2, T3, and so on. Specifically, to test the model’s consistency, it is expected that one model obtained using the dataset for T1 should provide good prediction results over T2, T3 and so on. Hence, pairwise comparisons are used. For each day (1, 2… 7), a survival model is built and then tested against the rest of the days. Additionally, the model estimates are compared to discover the variables that are consistently significant over the 7 days. To quantify the predictive accuracy, the Area Under a receiver operating characteristics Curve (AUC) is used to assess the results. This strategy has been considered a robust testing method and has been widely used in many different applications in survival analysis [14, 67, 68]. The AUC value ranges from 0 to 1. The higher the AUC value is, the better the prediction. An AUC value of 0.8 or above is considered to be excellent [69-72].

4 Result and discussion

4.1 Survival analysis results

The prediction results (AUCs) are summarized in Table 3. The research result shows good predictive performance. The research model identifies several important predictors, such as the efficiency measure (rEfficiency) and whether a sender previously contributed (Contri), see Table 4.

Table 3

Cross validated results on different dates (AUC)

	T1	T2	T3	T4	T5	T6	T7	Average
T1	\	0.826	0.828	0.89	0.811	0.808	0.844	0.835
T2	0.835	\	0.837	0.891	0.823	0.837	0.851	0.846
T3	0.835	0.834	\	0.896	0.827	0.823	0.854	0.845
T4	0.824	0.821	0.827	\	0.799	0.792	0.841	0.817
T5	0.83	0.833	0.841	0.888	\	0.828	0.851	0.845
T6	0.834	0.845	0.844	0.887	0.832	\	0.859	0.85
T7	0.836	0.834	0.841	0.887	0.826	0.824	\	0.841
Average	0.832	0.832	0.836	0.89	0.82	0.819	0.85	\
			Mean	Std				
			0.840	0.024				

The prediction results have an average AUC above 0.8 with a standard deviation of 0.024, indicating sound and stable accuracy over time. A model is built using one of the seven days' datasets to predict the acceptance of the revisions on each of the other days. The row averages of the AUC values reveal how well a single dataset's model predicts the other datasets.

The column averages show how well the other six days' models predict on one specific day. Individual difference is observed when comparing the column averages using Friedman's test [73], as *p* value < 0.001. It has been observed that in column average AUCs, T5 and T6 are ranked the lowest. One explanation for this difference is that T5 and T6 are weekends (September 5th – 6th 2015), and the sample sizes are relatively smaller and thus more difficult to predict. Similarly, T4 in the row averages yield the lowest in AUC to predict other datasets because the number of accepted cases are relatively smaller, subjected to the coming weekends. No other differences are observed in row average values, as Friedman's test returns *p* value = 0.077.

Table 4

Survival model estimates

*p < .05; **p < .01; ***p < .001
NA: sharedRepo found to be consistently 0 across all observations within some of the days
Hazard ratio shown below

Variables	T1	T2	T3	T4	T5	T6	T7
Sfer	1.006	0.99	0.988	0.989	1.007	0.977*	0.99
Sfol	0.999	0.994	0.99	0.979*	0.998	0.969**	0.978*
Sorg	0.997	0.993	0.981	0.866***	1.017	0.962	0.973
Srep	1.012***	1.011***	1.015***	1.011*	0.993	1.006	0.993
Ssub	0.993***	0.99***	0.982***	0.987**	0.993	0.986*	0.99**

Rfer	1.031***	1.033***	1.048***	1.059***	1.056***	1.039***	1.033***
Rfol	1	1.017***	1.028***	1.007	1.014	1.007	0.971***
Rorg	1.024***	1.026***	1.037***	1.052***	1.023**	1.006	1
Rrep	1.021**	1.017	1.07***	1.121***	1.026	0.961	1.075***
Rsub	0.98***	0.978***	0.968***	0.955***	0.965***	0.979***	0.978***
shareOrg	0.972	0.975	1.004	1.247*	0.985	1.2	0.875
shareRepo	0	Na	Na	Na	Na	0	3.14
shareSub	1.012**	1.032***	1.029***	1.063***	1.006	1.036*	1.018*
shareF	1.054***	1.077***	1.058***	1.101***	1.073***	1.118***	1.149***
sEffective	0.996	1.008	1.014*	1.023*	0.998	1.03*	1.02
sEfficiency	0.911*	0.901	0.919	1.133	0.915	0.89	0.884
sHierarchy	1.132	1.353**	1.065	1.168	1.531**	1.284	1.021
rEffective	0.986***	0.978***	0.958***	0.972***	0.975**	0.991	1.013
rEfficiency	3.25***	2.913***	2.95***	4.309***	1.953***	1.864**	3.754***
rHierarchy	2.945***	2.821***	2.195***	4.244***	2.955***	2.181***	2.642***
Contri	1.798***	2.311***	2.449***	1.812***	2.44***	2.739***	2.93***
Sy	0.927***	0.923***	0.952***	0.894***	0.955*	0.958	1.005
Ry	0.992	1.035**	0.978	0.979	0.979	1.044	1.002

Hazard ratios are reported in Table 4 for data collected from 7 different days. The receivers, who are in the pivotal role of accepting revisions, have 7 variables significantly associated with the acceptance in more than half of the datasets, while the senders have only 4. The senders' number of repositories and subscriptions are significant, but carry little weight to predict the outcome. When senders are previous contributors, the revisions are likely to be accepted at a minimum of 1.8 times faster. Receivers' followers, organizations, subscriptions, and repositories carry little weight to influence the outcome. However, the receivers' efficiency and hierarchy in their ego network have been found to significantly contribute to the acceptance of the revisions; the higher the value, the faster the revisions will be accepted. Receivers' effective size is likely to contribute as well, but the exact role is unknown since the results are inconsistent. Effective size varies among the egos because different ego networks have different sizes. A large effective size may be a result of a large network size, which takes more effort to maintain and is therefore not efficient. Hence, the effective size is likely to slow down acceptance, as found in the significant results shown in T1-T5. The number of shared friends and shared subscriptions to repositories contribute to the acceptance as well, but in a limited way.

In conclusion, the receiver's efficiency, hierarchy, and whether the sender was a previous contributor, are found to be the strong predictors. The other characteristics in general carry very limited weight in predicting the revision outcome.

The proportional hazard assumption is evaluated using the weighted residual test [64]. Time interaction terms are added to those variables that violated the proportional hazard assumption and compared with the original model as suggested from the literature [74, 75]. There are no major changes of significance and polarity of the coefficients signs regarding the variables tested in this study across the 7 datasets.

It is not surprising to find non-proportional variables in the datasets because networks on the social media platform are likely to change over time as users continue to interact with each other. However, researchers often empirically find that after correcting the proportional hazard assumption violation using the time varying coefficients, the new model estimation is similar to the original one [75, 76]. In conclusion, the model coefficients will change with time, but the changes are numerically insignificant in a week's follow-up time. Hence, we could assume they are approximately constant.

Further tests have been conducted to see if direction of the social network ties matter in predictions. The social network data is a directed network. The effect of the directions are tested by converting the directed network to an un-directed network. The social network then becomes a network of friendship ties, despite who follows who. The Cox proportional hazard model estimates are found to be consistent with the original model, as no major change of significance and polarity of the coefficients signs are found across the 7 datasets. The new un-directed model does not increase or decrease the predictive accuracy, as a Wilcoxon rank sum test returns p value = 0.8195 when comparing the AUC values of the un-directed network versus the directed one. Hence, the direction of the ties does not have an impact on the predictive results.

Finally, the Akaike information criterion (AIC) based stepwise model selection procedure [77] was deployed to find alternative models with better fit. However, the method did not lead to model results with significant improvements.

4.2 Implications

These research findings contribute to the literature in multiple aspects. This paper broadens the horizon of OSS research by exploring the time until acceptance issue in software revisions. Previously, software project size, knowledge domain, contributors' capability, and the developers' interaction [18-20] have been shown to influence the outcome of a revision. However, social network information is often neglected or studied with limited scope, e.g., focusing only on revision comments [20]. This paper investigates the problem with a different lens by addressing the structural aspect of the social networks. Formalized with the structural holes theory, repository owners' positional advantage has been found to be closely related to the time of acceptance.

The research findings distinguish the role of project contributor (the sender) and the maintainer (the receiver) during the revision process, which is composed of the submission and the review. Hence, the acceptance of a revision is not just based on how a sender programs, but also depends upon the receiver's effort to acknowledge the sender's contribution. As indicated in a previous study that interviewed the receivers [39], receivers as the project maintainers often find it difficult to assess the revisions. The receivers need to consider many different aspects including the quality of the submission, the coding style, and the general fit with other parts of the project. The primary focus is to assess the quality and the integrity to the whole project, regardless of whether the sender has a good record or belongs to a specific community. Receivers also note that assessing revision quality is a difficult task, as they have no prior knowledge to refer to, and it is currently not possible to automate the reviewing process to help them make the decision whether to accept in a timely manner. A sender's status, for instance organizations and networks are not a primary concern of the receiver. A sender's characteristics, such as number of years since registered as a user, might not be representative of the submission quality.

Given the fact that no specific features on both the submission and the sender could consistently support the decision, receivers often attempt to rely on other co-workers for review to make decisions, but other reviewers are often not available [39]. The receivers are in a position that requires sufficient experience in reading the code, understanding the functionality of the submission, thinking about the projects' general picture, and referring to co-workers' for support. These tasks require a lot of social resources, such as co-working experience, other people's opinions about the projects, and their input. A sender who has previously contributed lowers the demand of these resources, as the receiver has experience in dealing with the contributor. Thus, the sender having previously contributed reduces acceptance time. On top of the technical demands [19, 20, 40], a maintainer also has a managerial role, which relies on communication and experience to manage the product quality, the conversation with co-workers, and time.

Structural holes are related to various social resources in the networks. Revisions sent to certain receivers with structural holes in the network are found to be accepted 1.8 to 4.3 times more quickly than others. Structural holes in the receivers' network imply that they have access to different communities with different expertise; they are more experienced in working with different kinds of coders, with different types of coding style, functionalities, and so on. It is fair to imagine that they might also have more referees with different expertise to review the code. In total, receivers with structural holes have more resources to make faster decisions to accept. This explains why the structural holes measures of the receivers are more highly predictive than other measures such as the individual characteristics and the senders' networks. It is similar with some cases in other contexts where structural holes are related to higher profits in the competitive markets, or to faster promotion speed in organizations. In all these examples, people with structural holes in their networks benefit from more social resources to support their decision making.

The research results provide insight into various other distinct but relevant research questions. The study of software quality [78] is an example, since revisions sometimes improve the quality of the software. Likewise, project delay is widely studied in different industries [78-83]. Studying the delays in OSS development can glean new insights regarding an extended period for a repository to accept the revision. Collective software revisions in social media networks can be viewed as collaborative innovations. This paper extends the understanding of the collaboration network to the OSS community. Collaborative networking is often related to innovative product design [84-86], as it is believed that the unique features of a new product could be nourished by and harvest from the wisdom of the crowd [87]. Researchers are interested in de-centralized and self-organized teams' performance on innovative tasks [88]. In practice, it is often difficult to apply such organizational changes to reach the full potential for pursuing the innovation. This study shows that the OSS could be a pivotal spot to study innovation through the de-centralized nature of the social networks. While the repositories are hosted by their owners, contributors from other parts of the world can send innovative revisions to add value.

From a methodological perspective, the contribution of this paper is the use of social network analysis in conjunction with survival analysis to study the "time to accept" problem. The structural holes theory has served as a conductor to operationalize the social network data for survival analysis. Distinct from canonical regression methods, as seen in past works [18, 20], the survival model uses follow-up data samples to estimate the weights of the variables in order to study the time to acceptance. Additionally, the survival model is capable of using censored data. This mix yields a novel analytical approach that is based on profound social network theories. The research model's predictive performance was validated with real-life data from social media platform [89, 90].

Practitioners may find the research results interesting, as the research indicates a quantitative method to support the decision making in software development by forecasting revision acceptance. This allows project managers to deal with the potential risks by considering the project delay and maturity. Besides the research methods, the research findings suggest possible managerial actions to assist with the software development process. Software project team managers should not underestimate the value of social network ties, which link to the successful integration of contributions from external and internal programmers. Managers should realize that the social networks are valuable resources to investigate the different development and revision time of software projects. Significant results of receivers' structural holes variables indicate the importance of interacting with different programmers in social networks. In order to facilitate the successful integration of the software revision to the project, managers should acquire skills to efficiently distribute their ties to the developers. Acting as boundary spanners, managers should learn to work with different groups of software developers internally and externally. For instance, companies could organize offline and online events, workshops, and other activities to provide opportunities for their managers to approach external software developers more easily. Adopting different communication channels could enable managers to connect broadly with the community and benefit from the diverse knowledge and expertise of developers.

In this study, network data derived from social media platform is found to be useful to understand collaboration, without using domain-specific terminologies such as the technical features. This allows the findings to be transferred to manage the collaboration in many other domains. Collaborative product design teams should learn to leverage tools such as social media to engage contributors outside of organizations. With the changing nature of the market, only products with unique features appreciated by the public will stand out. Although this paper primarily studies the OSS community, closed-source organizations can also benefit, since increasingly more closed-source companies are using OSS, and the OSS working style is influencing the managerial style in other organizations [3, 6, 22, 24]. GitHub revisions are active during weekdays with very predictive outcomes indicated in Table 3. It seems possible to integrate the GitHub usage seamlessly during working days in companies. Furthermore, though the research focuses on software collaboration, websites such as GitHub would also be a good place for general-purpose collaboration. Online users might work jointly on a document despite not being programmers [91].

The impact of studying the time to accept a revision is meaningful not just for software engineering but also for studying group work in general. Social media networks have the potential to contribute to organizations in different aspects including performance improvement, task distribution and information sharing. From a human resource perspective, network position gives an additional indicator to project "teamness"— the team-oriented nature of a candidate and his/her ability to contribute. It also implies that individuals, such as programmers, may want to market themselves better through participating actively in their community by submitting contributions.

4.3 Limitations and future work

The scope of the paper is limited to user-user networks. The co-ownership of the repositories and co-membership in organizations could be the alternative sources of social networks to extend the current study. However, working with multiple types of objects in the social networks can be cumbersome, as different types of network have different underlying assumptions [92]. This work mainly considers the social network structure among the users. It is possible to extend the study by integrating the multilevel effect, such as the organizations and the subscriptions.

5 Conclusion

The research presented in this paper studied the role of social networks to predict software revision outcome from the online OSS host GitHub. A data sample of 32,962 revisions, 20,399 software projects, and the social network of 234,322 users was collected. Research methods such as survival analysis and social network analysis were used to explore the research topic. The approach was tested with statistical methods to ensure the stability and generalization of the results. The study obtained good prediction accuracy with an average AUC of 0.84. Research findings pointed out that positional advantage in social networks is closely related to the faster acceptance of the revision. The paper deepens the understanding of the software revision process. The social networks play a role in the OSS development, tightly related to quality management, project duration, and collaborative innovation. The research outcome further suggests that social media provides vital information that could support the decisions in the organization regarding managerial practice and product design.

Acknowledgements

The authors thank the editors and three anonymous reviewers for their encouragement, careful reading of the manuscript and helpful comments.

References

[1] G. Canfora, L. Cerulo, M. Di Penta, Tracking Your Changes: A Language-Independent Approach, Software, IEEE, 26 (2009) 50-57.
[2] B. Westfechtel, B.P. Munch, R. Conradi, A layered architecture for uniform version management, Software Engineering, IEEE Transactions on, 27 (2001) 1111-1133.
[3] K.-J. Stol, B. Fitzgerald, Inner Source--Adopting Open Source Development Practices in Organizations: A Tutorial, IEEE Software, (2015) 60-67.
[4] M. Bloch, S. Blumberg, J. Laartz, Delivering large-scale IT projects on time, on budget, and on value, mckinsey & company, mckinsey.com, 2012.
[5] T. Dixon, I. Melve, R. Meneses, T. Verschuren, Building large-scale information services: tools and experiences from the DESIRE project, Computer Networks and ISDN Systems, 30 (1998) 1559-1569.
[6] R. Schuwer, M. van Genuchten, L. Hatton, On the Impact of Being Open, Software, IEEE, 32 (2015) 81-83.
[7] M. Trier, A. Richter, The deep structure of organizational online networking-an actor-oriented case study, Information Systems Journal, (2014).
[8] J. Klier, M. Klier, R.T. Wigand, The connectedness, pervasiveness and ubiquity of online social networks, Computer Networks, (2014) 473-476.
[9] T.A. Sykes, V. Venkatesh, S. Gosain, Model of acceptance with peer support: A social network perspective to understand employees' system use, Mis Quarterly, (2009) 371-393.
[10] W. Bolander, C.B. Satornino, D.E. Hughes, G.R. Ferris, Social Networks Within Sales Organizations: Their Development and Importance for Salesperson Performance, Journal of Marketing, 79 (2015) 1-16.
[11] M. Girolami, S. Chessa, A. Caruso, On service discovery in mobile social networks: Survey and perspectives, Computer Networks, 88 (2015) 51-71.
[12] K. Zhu, W. Li, X. Fu, L. Zhang, Data routing strategies in opportunistic mobile social networks: Taxonomy and open challenges, Computer Networks, 93, Part 1 (2015) 183-198.
[13] J. Peng, Y. Zhu, W. Shu, M.-Y. Wu, When data contributors meet multiple crowdsourcers: Bilateral competition in mobile crowdsourcing, Computer Networks, 95 (2016) 1-14.
[14] T. Paul, A. Famulari, T. Strufe, A survey on decentralized Online Social Networks, Computer Networks, 75, Part A (2014) 437-452.
[15] S. Behrendt, A. Richter, M. Trier, Mixed methods analysis of enterprise social networks, Computer Networks, 75, Part B (2014) 560-577.
[16] J. Putzke, K. Fischbach, D. Schoder, P.A. Gloor, Cross-cultural gender differences in the adoption and usage of social media platforms – An exploratory study of Last.FM, Computer Networks, 75, Part B (2014) 519-530.
[17] K. Muthukumaran, A. Choudhary, N. Murthy, Mining GitHub for Novel Change Metrics to Predict Buggy Files in Software Systems, Computational Intelligence and Networks (CINE), 2015 International Conference on, IEEE, 2015, pp. 15-20.
[18] G. Gousios, M. Pinzger, A.v. Deursen, An exploratory study of the pull-based software development model, Proceedings of the 36th International Conference on Software Engineering, ACM, 2014, pp. 345-355.
[19] D.M. Soares, M.L. de Lima Júnior, L. Murta, A. Plastino, Acceptance factors of pull requests in open-source projects, Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM, 2015, pp. 1541-1546.
[20] Y. Yu, H. Wang, V. Filkov, P. Devanbu, B. Vasilescu, Wait for it: Determinants of pull request evaluation latency on GitHub, Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on, IEEE, 2015, pp. 367-371.

[21] M. Michlmayr, B. Fitzgerald, K.-J. Stol, Why and How Should Open Source Projects Adopt Time-Based Releases?, Software, IEEE, 32 (2015) 55-63.
[22] K. Matsudaira, Lean software development: building and shipping two versions, Communications of the ACM, 58 (2015) 56-58.
[23] J.D. Herbsleb, A. Mockus, An empirical study of speed and communication in globally distributed software development, Software Engineering, IEEE Transactions on, 29 (2003) 481-494.
[24] D. Riehle, How Open Source Is Changing the Software Developer's Career, Computer, 48 (2015) 51-57.
[25] T. Fawcett, An introduction to ROC analysis, Pattern Recogn. Lett., 27 (2006) 861-874.
[26] P.J. Heagerty, Y. Zheng, Survival model predictive accuracy and ROC curves, Biometrics, 61 (2005) 92-105.
[27] I.H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann 2005.
[28] T. Hastie, R. Tibshirani, J. Friedman, J. Franklin, The elements of statistical learning: data mining, inference and prediction, The Mathematical Intelligencer, 27 (2005) 83-85.
[29] J.W. Paulson, G. Succi, A. Eberlein, An empirical study of open-source and closed-source software products, Software Engineering, IEEE Transactions on, 30 (2004) 246-256.
[30] T. Gyimothy, R. Ferenc, I. Siket, Empirical validation of object-oriented metrics on open source software for fault prediction, Software Engineering, IEEE Transactions on, 31 (2005) 897-910.
[31] L. Dabbish, C. Stuart, J. Tsay, J. Herbsleb, Social coding in GitHub: transparency and collaboration in an open software repository, Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, ACM, 2012, pp. 1277-1286.
[32] V.D. Bianco, L. Lavazza, S. Morasca, D. Taibi, A survey on open source software trustworthiness, Software, IEEE, 28 (2011) 67-75.
[33] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, V. Filkov, Quality and productivity outcomes relating to continuous integration in GitHub, Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ACM, Bergamo, Italy, 2015, pp. 805-816.
[34] N. Ning, S. Kumar, Joint Effect of Team Structure and Software Architecture in Open Source Software Development, Engineering Management, IEEE Transactions on, 60 (2013) 592-603.
[35] K. Chaykowski, Facebook Open-Sources Artificial Intelligence Hardware Design For The First Time, forbes tech, 2015
[36] D.A. Grier, The GitHub Effect, Computer, 48 (2015) 116-116.
[37] F. Thung, T.F. Bissyandé, D. Lo, L. Jiang, Network structure of social coding in github, Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on, IEEE, 2013, pp. 323-326.
[38] A. Begel, J. Bosch, M.-A. Storey, Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder, Software, IEEE, 30 (2013) 52-66.
[39] G. Gousios, A. Zaidman, M.-A. Storey, A. Van Deursen, Work practices and challenges in pull-based development: The integrator's perspective, Proceedings of the 37th International Conference on Software Engineering-Volume 1, IEEE Press, 2015, pp. 358-368.
[40] J. Tsay, L. Dabbish, J. Herbsleb, Influence of social and technical factors for evaluating contribution in GitHub, Proceedings of the 36th international conference on Software engineering, ACM, 2014, pp. 356-366.
[41] S.P. Borgatti, M.G. Everett, J.C. Johnson, Analyzing social networks, SAGE Publications Limited 2013.
[42] A. Sundararajan, F. Provost, G. Oestreicher-Singer, S. Aral, Information in Digital, Economic and Social Networks Information Systems Research, Forthcoming (2012).
[43] S. Wasserman, K. Faust, Social network analysis: Methods and applications, Cambridge university press 1994.
[44] M. McPherson, L. Smith-Lovin, J.M. Cook, Birds of a Feather: Homophily in Social Networks, Annual Review of Sociology, 27 (2001) 415-444.
[45] C.C. Aggarwal, Social Network Data Analytics, Springer Publishing Company, Incorporated 2011.
[46] G.C. Kane, M. Alavi, G. Labianca, S.P. Borgatti, What's different about social media networks? A framework and research agenda, MIS Quarterly, 38 (2014) 275-304.
[47] J. Mcauley, J. Leskovec, Discovering social circles in ego networks, ACM Transactions on Knowledge Discovery from Data (TKDD), 8 (2014) 4.

[48] A.G. J. Harkola, Diffusion of technology: cohesion or structural equivalence?, Academy of Management Meeting., Vancouver (1995) 5.
[49] R.S. Burt, Social Contagion and Innovation: Cohesion versus Structural Equivalence, American Journal of Sociology, 92 (1987) 1287-1335.
[50] W. Pan, N. Aharony, A. Pentland, Composite Social Network for Predicting Mobile Apps Installation, AAAI, 2011.
[51] R.S. Burt, Positions in Networks, Social Forces, 55 (1976) 93-122.
[52] R.S. Burt, Structural holes: The social structure of competition, Harvard university press 2009.
[53] R.S. Burt, Structural holes and good ideas1, American journal of sociology, 110 (2004) 349-399.
[54] A. Zaheer, G. Soda, Network evolution: The origins of structural holes, Administrative Science Quarterly, 54 (2009) 1-31.
[55] P.D. Allison, Survival Analysis Using SAS®: A Practical Guide, Second Edition, SAS Institute 2010.
[56] J.D. Kalbfleisch, R.L. Prentice, The statistical analysis of failure time data, John Wiley & Sons 2011.
[57] D.G. Kleinbaum, M. Klein, Survival analysis: a self-learning text, 3rd ed., Springer Science & Business Media 2012.
[58] S. Park, J. Lee, M. Lee, Sustaining Web 2.0 services: A survival analysis of a live crowd-casting service, Decis. Support Syst., 54 (2013) 1256-1268.
[59] R. Kauffman, A. Techatassanasoontorn, B. Wang, Event history, spatial analysis and count data methods for empirical research in information systems, Inf Technol Manag, 13 (2012) 115-147.
[60] S.P. Borgatti, Centrality and network flow, Social networks, 27 (2005) 55-71.
[61] D.R. Cox, Regression models and life tables, JR stat soc B, 34 (1972) 187-220.
[62] Y. Wax, S.G. Baker, B.H. Patterson, A score test for non-informative censoring using doubly sampled grouped survival data, Applied statistics, (1993) 159-172.
[63] X. Huang, R.A. Wolfe, C. Hu, A test for informative censoring in clustered survival data, Statistics in medicine, 23 (2004) 2089-2107.
[64] P.M. Grambsch, T.M. Therneau, Proportional hazards tests and diagnostics based on weighted residuals, Biometrika, 81 (1994) 515-526.
[65] O.S. Collaboration, Estimating the reproducibility of psychological science, Science, 349 (2015) aac4716.
[66] V. Venkatesh, M.G. Morris, G.B. Davis, F.D. Davis, User acceptance of information technology: toward a unified view, MIS Q., 27 (2003) 425-478.
[67] H.-C. Chen, R.L. Kodell, K.F. Cheng, J.J. Chen, Assessment of performance of survival prediction models for cancer prognosis, BMC medical research methodology, 12 (2012) 102.
[68] L. Duchateau, P. Janssen, The frailty model, Springer Science & Business Media 2007.
[69] R. Ohlemüller, E.S. Gritti, M.T. Sykes, C.D. Thomas, Quantifying components of risk for European woody species under climate change, Global Change Biology, 12 (2006) 1788-1799.
[70] W. Zhu, N. Zeng, N. Wang, Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS® implementations, NESUG proceedings: health care and life sciences, Baltimore, Maryland, (2010) 1-9.
[71] R. Tao, X. Huang, J. Wang, H. Zhang, Y. Zhang, M. Li, Proposed diagnostic criteria for internet addiction, Addiction (Abingdon, England), 105 (2010) 556-564.
[72] S. Pittman, J. Christensen, C. Caldow, C. Menza, M. Monaco, Predictive mapping of fish species richness across shallow-water seascapes in the Caribbean, ecological modelling, 204 (2007) 9-21.
[73] M. Hollander, D.A. Wolfe, E. Chicken, Nonparametric statistical methods, John Wiley & Sons 2013.
[74] T.M. Therneau, P.M. Grambsch, Modeling survival data: extending the Cox model, Springer Science & Business Media 2000.
[75] L. Thomas, E.M. Reyes, Tutorial: Survival Estimation for Cox Regression Models with Time-Varying Coefficients Using SAS and R, 2014, 61 (2014) 23.
[76] I. Persson, Essays on the assumption of proportional hazards in Cox regression, Acta Universitatis Upsaliensis 2002.
[77] F. Kemp, Modern Applied Statistics with S, Journal of the Royal Statistical Society: Series D (The Statistician), 52 (2003) 704-705.
[78] P. Dowling, K. McGrath, Using free and open source tools to manage software quality, Communications of the ACM, 58 (2015) 51-55.
[79] Z. Ahmedshareef, R. Hughes, M. Petridis, Exposing the Influencing Factors on Software Project Delay with Actor-Network Theory, Electronic Journal of Business Research Methods, 12 (2014).

[80] M. Gunduz, Y. Nielsen, M. Ozdemir, Fuzzy assessment model to estimate the probability of delay in Turkish construction projects, Journal of Management in Engineering, (2013) 04014055.

[81] M. Ruqaishi, H.A. Bashir, Causes of delay in construction projects in the oil and gas industry in the gulf cooperation council countries: a case study, Journal of Management in Engineering, 31 (2013) 05014017.

[82] J. Pfeifer, K. Barker, J.E. Ramirez-Marquez, N. Morshedlou, Quantifying the risk of project delays with a genetic algorithm, International Journal of Production Economics, 170 (2015) 34-44.

[83] N.S.N. Rao, N. Jigeesh, Analysis and control of issues that delay pharmaceutical projects, Business: Theory and Practice/Verslas: Teorija ir Praktika, 16 (2015) 252-263.

[84] J. Kratzer, C. Lettl, A social network perspective of lead users and creativity: An empirical study among children, Creativity and Innovation Management, 17 (2008) 26-36.

[85] J. Kratzer, C. Lettl, N. Franke, P.A. Gloor, The social network position of lead users, Journal of Product Innovation Management, (2015).

[86] R.T. Leenders, W.A. Dolfsma, Social Networks for Innovation and New Product Development, Journal of Product Innovation Management, (2015).

[87] P. Gloor, Coolfarming: Turn Your Great Idea Into the Next Big Thing, AMACOM Div American Mgmt Assn 2010.

[88] P.A. Gloor, P. Margolis, M. Seid, G. Dellal, Coolfarming–Lessons from the Beehive to Increase Organizational Creativity, Available at SSRN 2461532, (2014).

[89] P. Gang, M. Jifeng, Network Structures and Online Technology Adoption, Engineering Management, IEEE Transactions on, 58 (2011) 323-333.

[90] G. Shmueli, To Explain or to Predict?, Statistical Science 25 (2010) 20.

[91] R. Mcmillan, From Collaborative Coding to Wedding Invitations: GitHub Is Going Mainstream, WIRED, WIRED Online, 2013.

[92] T.A.B. Snijders, A. Lomi, V.J. Torló, A model for the multiplex dynamics of two-mode and one-mode networks, with an application to employment preference, friendship, and advice, Social Networks, 35 (2013) 265-276.