Can Developer-Module Networks Predict Failures?

Martin Pinzger [†]
Department for Informatics
University of Zurich
Zurich, Switzerland
pinzger@ifi.uzh.ch

Nachiappan Nagappan
Software Reliability Research
Microsoft Research
Redmond, USA
nachin@microsoft.com

Brendan Murphy
System Dependability
Microsoft Research
Cambridge, UK
bmurphy@microsoft.com

ABSTRACT
Software teams should follow a well defined goal and keep their work focused. Work fragmentation is bad for efficiency and quality. In this paper we empirically investigate the relationship between the fragmentation of developer contributions and the number of post-release failures. Our approach is to represent developer contributions with a developer-module network that we call contribution network. We use network centrality measures to measure the degree of fragmentation of developer contributions. Fragmentation is determined by the centrality of software modules in the contribution network. Our claim is that central software modules are more likely to be failure-prone than modules located in surrounding areas of the network. We analyze this hypothesis by exploring the network centrality of Microsoft Windows Vista binaries using several network centrality measures as well as linear and logistic regression analysis. In particular, we investigate which centrality measures are significant to predict the probability and number of post-release failures. Results of our experiments show that central modules are more failure-prone than modules located in surrounding areas of the network. Results further confirm that number of authors and number of commits are significant predictors for the probability of post-release failures. For predicting the number of post-release failures the closeness centrality measure is most significant.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Software Metrics – complexity measures, performance measures, process metrics, product metrics

General Terms
Management, Measurement, Reliability, Experimentation

Keywords
Failure Prediction, Social Network Analysis, Developer Contribution Network, Network Centrality Measures

1. INTRODUCTION
Building successful software systems is a team effort. A number of best practices on how to manage teams, coordinate working efforts, and work towards a common goal has been reported in literature, for example in [7] and [8]. Fragmentation is bad for team formation and efficiency [8]. In our previous study with the Microsoft Windows Vista project we provided empirical evidence that organizational structures including fragmentation, for example of module ownership, affect software quality [34]. In this paper we further investigate the effects of fragmentation by *analyzing the relationships between the number and structure of contributions from developers to a software module and its number of post-release failures*.

Our approach is to represent contributions of software modules in a graph that we call contribution network. Nodes in the contribution network represent modules and developers. Edges represent contributions of developers to software modules. The network is computed of change log data obtained from the version control system of a software project. The centrality of a software module in the contribution network is used as a measure for the fragmentation of developer contributions. Basically, central software modules have many contributions from various developers.

Our claim is that *central modules are more likely to be failure-prone than modules located in surrounding areas of the contribution network*. In our study, modules are Microsoft Windows binaries, such as .exe's and .dll's. Based on the claim we state the following three research hypotheses:

H1 Network centrality measures are significant indicators for failure-prone binaries.

H2 The centrality of binaries in the contribution network correlates positively with the number of post-release failures—the more central a binary the more post-release failures it will have.

H3 Advanced network centrality measures significantly improve failure prediction models.

We investigated these hypotheses with the Microsoft Windows Vista project. For the validation of the hypotheses we use linear and logistic regression analysis. Regression relates network centrality measures of binaries with the number of post-release failures (reported within the six month after the first release of Vista). Results of our empirical study show a strong correlation between the centrality of software modules and the number of post-release failures. The basic centrality measures *number of authors* and *number of commits* can predict on average more than 83% of failure-prone Vista binaries. This confirms results of previous studies, such as, presented in [15], [28], and [41]. *Closeness centrality* in combination with number of authors and

[†] Martin Pinzger was a visiting researcher with the Software Reliability Research Group, Microsoft Research in the fall of 2007 when this work was carried out.

number of commits can predict the number of post-release failures of Vista binaries. Including the closeness centrality measures improves the R-Square values of linear regression models by 0.326 on average and the standard error of estimates. This denotes a significant improvement compared to existing prediction models. Furthermore, it provides significant empirical evidence for keeping teams focused on a set of binaries to prevent post-release failures and improve software quality.

The remainder of the paper is structured as follows: Section 2 describes the contribution network which we establish from change log information. Network centrality measures are presented in Section 3 followed by the empirical study in Section 4. We present related work in Section 5. Section 6 draws the conclusions and indicates future work.

2. CONTRIBUTION NETWORK

In our approach we represent the associations between developers and software modules (binaries) with a contribution network. In social network theory such networks are called association networks [40].

A contribution network is an undirected bipartite graph G that is formally defined as $G = (D, B, E)$. D and B are the two sets of vertices, that represent the two partitions of the graph, and E is a set of edges between vertices $E \subseteq \{(d, b) | d \in D \wedge b \in B\}$. For our experiments, D represents the set of developers and B the set of binaries. An edge $e \in E$ denotes a contribution of a developer $d \in D$ to a binary $b \in B$. A contribution refers to a commit of a developer to a binary in the versioning system. We limit edges to always be between a developer and a binary and there are no self-loops (i.e., binaries cannot contribute to itself nor can developers). Edges in the contribution network are undirected to allow the navigation of contributions in both directions. Edge weights are used to denote the number of commits a developer did to a binary.

For creating the contribution network we use the change log information stored in versioning systems. Each log entry contains information about which developer committed a change to which source file, and when. From the build environment we obtain the information which source file(s) are compiled to which binary. For each author we sum up the number of commits to source files of binaries. The author, binary, and the number of commits of an author to a binary represent a contribution record that is used to create the network.

Figure 1 depicts an example of a contribution network. Circles represent developers, rectangles represent binaries, and edges represent developer contributions to binaries. Edge labels denote the number of commits of developers to binaries. For example, developer $Alice$ committed 6 times to binary a.

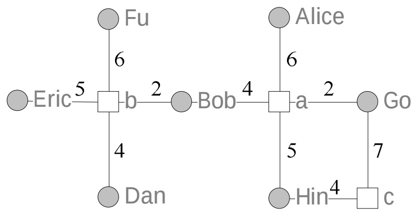


Figure 1. Contribution network example with contributions of 7 developers to 3 binaries.

Our analysis concentrates on predicting the failure-proneness of binaries. Failure-proneness refers to the likelihood of a binary to encounter post-release failures. Referring to the contribution network we claim that *central binaries are more likely to be failure-prone than binaries located in surrounding areas of the network*.

In the contribution network the centrality of a binary is influenced by the number of developers that worked on it and the number of commits. Concerning the number of developers and number of commits of our example network the binaries a and b are more central than binary c. Binary a and b both got 17 commits from 4 developers. Binary c got 11 commits from 2 developers. Therefore, binary a and b are more likely to have failures than binary c.

The extent to which a developer concentrates his/her working efforts on a set of binaries is another influencing factor to the centrality of binaries. Centrality increases if a developer works on many other binaries. In contrast, centrality decreases if developers retain focus on a set of binaries. Referring to our example in Figure 1, we expect binary a to be more failure-prone than binary b because the contributions to binary b are more *focused* than the contributions to binary a. Developers Dan, $Eric$, and Fu concentrated their working efforts on binary b, i.e., they did not contribute to any other binaries. Only Bob contributed also to binary a. In contrast, the sub-graph for binary a shows that 3 out of 4 developers worked also on the other two binaries. Only developer $Alice$ concentrated her work on binary a solely. Refining our interpretation from before we expect binary a to be more failure-prone than binary b. Therefore, from the set of three binaries we expect a to be the most failure-prone binary followed by binary b. Binary c is expected to be the least failure-prone binary.

Our motivation for using centrality as a predictor for failure-proneness is based on results of our previous studies [34]. Results showed that organizational metrics were statistically significant predictors of failure-proneness. In addition, our work is motivated by related approaches that empirically validated the significance of measures computed of change log information to predict defects and failures, for example [15], [23], [25], and [33]. In extension to existing approaches, we argue that number of commits and number of authors provide only a limited view on developer contributions and its effects on software quality. As in other manufacturing processes high product quality needs developers to work in teams and keep their work focused.

In the next section, we present the different network centrality measures that we use to measure the centrality of binaries in a contribution network.

3. NETWORK CENTRALITY MEASURES

We use centrality measures that stem from social network theory to measure the centrality of binaries in a contribution network. In social network analysis, there exist several meanings and algorithms to compute centrality, namely *Degree*, *Closeness*, and *Betweenness* [17]. In this section we explain the set of centrality measures that we used in our experiments. For a more comprehensive overview of social network methods and applications we refer the reader to text books, such as [17] and [40].

For the explanation of the different network centrality measures we use the social network term *actor* to refer to a binary or developer. The term *tie* is used when referring to a developer contribution (i.e., an edge in the contribution network). In general, we compute network centrality measures for both, binary and developer nodes, in the contribution network. Concerning the analysis the focus is on the centrality of binaries.

For the measurement we use the Ucinet tool [6] which provides a number of social network centrality measures. Most of the centrality measures are computed with the dichotomized adjacency matrix of a network graph. In a dichotomized network graph, weights of edges are mapped to either 0 (no contribution) or 1 (at least one contribution). Where appropriate and supported by Ucinet we also compute the centrality with weighted ties.

We demonstrate each centrality measure with the example contribution network presented in Figure 1. Measured values are mapped to the size of rectangles and circles. High centrality of a binary or developer is represented by large rectangles and circles respectively. The resulting graphs for the different centrality measures are depicted in Figure 2 (for edge weights see Figure 1).

3.1 Degree Centrality

Degree centrality measures are computed based on the number of ties that an actor has. Basically, the more ties an actor has the more central is the actor. We use two different degree centrality measures which are:

Freeman degree centrality [13]: This measure considers centrality as the number of ties an actor has with other actors. When applied to a dichotomized network graph this refers to the number of developers that contributed to a binary. We call this measure *nrAuthors*. When applied to a network with weighted ties the Freeman degree denotes the number of commits to a binary. We call this measure *nrCommits*. These are two traditional measures that have been used in a number of previous experiments (e.g., [15], [28], [41]). Although they turned out to be significant predictors, we argue that these two measures provide a limited view on the structures of developer contributions. They measure centrality of a binary based on its direct neighborhood without considering the contributions of its developers to other binaries.

The result of this centrality measure is depicted by Figure 2 (a). Binary *a* and *b* both got 17 commits from 4 different developers marking these two binaries equally central in the contribution network. The corresponding rectangles are of equal size. Binary *c* is the least central binary.

Bonacich's power [5]: Bonacich measures centrality of an actor based on the centrality of the connected other actors. On the one hand, an actor is central if it is connected to actors that have connections to *many* other actors. On the other hand, an actor is powerful if it is connected to actors that have connections to *few* other actors. The algorithm can be configured by the Beta parameter. A positive Beta is used to calculate centrality and a negative Beta is used to calculate power. We focus on centrality and use the measure calculated with a positive Beta. Furthermore, we compute Bonacich's power of dichotomized networks as well as of networks with weighted ties. The two corresponding measures are *dPower* and *Power*.

Applying this measurement with a Beta of +0.2[1] to the dichotomized example network yields the graph depicted in Figure 2 (b). The same measure applied to the network with weighted ties with a Beta of +0.5 yields the graph depicted in Figure 2 (c). According to Figure 2 (b) binary *a* is the most central binary with a centrality of 7.08 followed by binary *b* with 6.29 and *c* with 3.66. Three out of four developers of binary *a* contributed also to other binaries.

[1] Calculated with the Ucinet tool

The graph in Figure 2 (c) shows a similar picture. With a power of 32.1 binary *a* is most central followed by the binaries *b* with 27.7 and *c* with 22.3. Binary *c* is rated more central than in the dichotomized network because of the relative high number of contributions it got from developer *Hin* and *Go*.

3.2 Closeness Centrality

Closeness centrality emphasizes the distance of an actor to other actors in the network. In this paper we use two such node distance measures:

Freeman approach with geodesic paths [13]: This approach measures the centrality of an actor by accumulating the lengths of shortest paths (i.e., geodesic paths) from an actor to all other actors in the network. The sum denotes the farness of an actor and its reciprocal is called closeness. The closer the other actors are to an actor the more central an actor is. Closeness centrality is strongly influenced by the degree of connectivity of a network. If all nodes are connected with each other the closeness centrality of each actor in the network is minimal. Concerning the contribution network the more developers contribute to other binaries the higher is the connectivity and the closer binaries are to each other. The measurement is applied to the dichotomized contribution network and the normalized closeness measure is called *ndCloseness*. Normalization is by the minimal possible closeness.

In the example network binary *a* is closest to all other nodes with a closeness centrality of 52.94 followed by *b* with 47.37 and *c* with 21.03. The size of rectangles in Figure 2 (d) highlights the corresponding differences in closeness centrality.

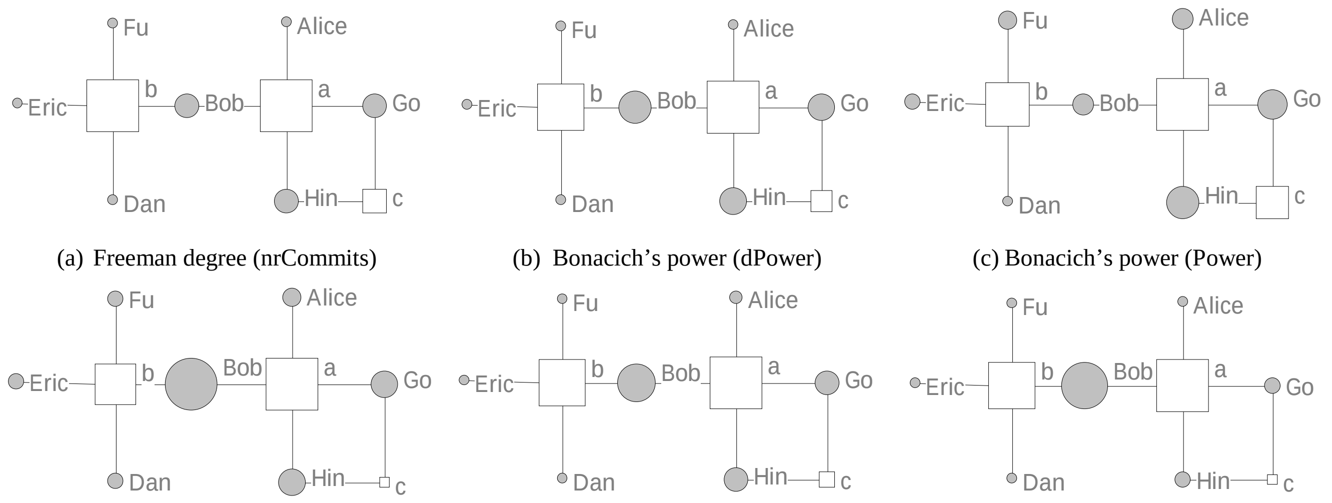
Reachability [6]: Similar to closeness this measure also strongly depends on the connectivity of a network. Reachability denotes the portion of other actors in the network that starting from an actor can be reached within 1, 2, …, n steps. Actors close to an actor are weighted higher than actors further away. Values are normalized by the number of actors. Binaries with contributions from many developers typically have high reachability in the network, therefore are more likely to have post-release failures. We compute reachability based on the dichotomized network and the resulting normalized measure is *ndReach*.

Concerning our example this measure yields similar results as *ndCloseness*: binary *a* is most central with a reachability of 0.7 followed by binary *b* with 0.675 and *c* with 0.502. Figure 2 (e) depicts the corresponding graph.

3.3 Betweenness Centrality

Betweenness centrality denotes the extent to which information flows through an actor to get from one actor to another actor. The more information flows through an actor the higher is its betweenness centrality. For this centrality we use one measure, which is:

Freeman node betweenness [13]: The more connections between actors go through an actor, the more power this actor has. If, however, there are alternative paths for actors to make connections to other actors, the actor loses some of its power. Ucinet computes the betweenness for dichotomized networks and the resulting normalized measure is called *ndBetweenness*. Normalization is by the maximum possible betweenness (all



(a) Freeman degree (nrCommits) (b) Bonacich's power (dPower) (c) Bonacich's power (Power)

(d) Freeman closeness (ndCloseness) (e) Reachability (ndReach) (f) Freeman betweenness (ndBetweenness)

Figure 2. Example contribution network with results of different network centrality measures mapped to the size of nodes. Circles denote developers, rectangles denote binaries, and edges denote contributions.

actors are connected by a single actor). A binary has a high betweenness centrality if many developers contribute to it and but do not work on many other binaries in common. On the one hand, this means that the work on this binary is focused. On the other hand developers do not work together on other binaries. High betweenness centrality, therefore, can mean both, low and high fragmentation and there is no clear consensus on the effect of this measure, yet. In our example network the binary with the highest betweenness centrality is *a* with 65.28 followed by *b* with 58.33 and *c* with 1.39. The graph in Figure 2 (f) clearly shows that there are not many alternative ways around the binaries *a*, *b*, and developer *Bob*. That is why the betweenness centrality of these nodes is high.

3.4 Summary

Network centrality measures, except Freeman degree centrality, compute centrality of an actor based on the whole contribution network. Freeman degree centrality takes into account only the direct neighborhood of an actor. In the remainder of the paper, we refer to Freeman degree centrality measures *nrAuthors* and *nrCommits* as basic centrality measures. The other centrality measures *dPower, Power, ndCloseness, ndReach,* and *ndBetweenness* are referred to as advanced centrality measures.

The differences between basic and advanced centrality measures are demonstrated by the graphs in Figure 2. While Freeman degree centrality obtains binaries *a* and *b* as equally central the advanced centrality measures yield binary *a* as the most central binary in the contribution network followed by binary *b*. This corresponds to our notion of fragmentation of developer contributions which for binary *a* is higher than for binary *b*. The centrality of binary *c* is significantly lower for each measurement, basically, because it got contributions from only two developers.

In this section we have discussed several network centrality measures and have explained them with examples in the context of our empirical study. The mathematical computation details are beyond the scope of this paper and are discussed in standard social network books, such as [17] and [40]. We focus more on the applications of these measures to our experiment. In the next section we explore the relationship between the different centrality measures and the number of post-release failures with the Microsoft Windows Vista binaries.

4. EMPIRICAL STUDY WITH VISTA

Microsoft Windows Vista comprises 3500+ binaries with contributions from a few thousand developers. Change log data was collected from beginning of the project to the first release of Vista. For each binary we computed the number of commits per author. The number of post-release failures is calculated by counting the number of failures reported (if any) for a binary within the first six months after shipping the first release of Vista.

During a first inspection of the data we found a small number of authors with contributions to almost all the binaries. Some of them were key-contributors others, however, were so called horizontal authors, or both. Horizontal authors worked, for instance, on fixing static source code defects in the build lab, ran instrumentations on the code, etc., each time resulting in a slight change in the source code of a binary. Each such change led to additional change log entries and consequently additional ties in the contribution network. To reduce the impact of horizontal authors we filtered out their minor contributions to binaries. For Vista, we obtained a contribution network that was fully connected. This clearly indicated that the interaction on binaries was not local which further motivated our experiments.

We integrated the results and used the statistical package SPSS to investigate hypotheses H1-H3 stated in the introduction. In the following we first report on the results of the correlation analysis. Hypothesis H1—network centrality measures are significant indicators for failure-prone binaries—is investigated in Section 4.2. Hypothesis H2—the centrality of binaries in the contribution network correlates positively with the number of post-release failures—is investigated in Section 4.3. Hypothesis H3—advanced network centrality measures significantly improve failure prediction models—is investigated in Section 4.4. In Section 4.5 we summarize the results and discuss our findings.

Table 1. Spearman correlation between the number of post-release failures and network centrality measures of Microsoft Windows Vista binaries. Values greater 0.7 denote strong correlations. All correlations are significant at the 0.01 level (2-tailed).

	nrFailures	nrCommits	nrAuthors	Power	dPower	ndCloseness	ndReach	ndBetweenness
nrFailures	1.000	0.700	0.699	0.692	0.740	0.747	0.746	0.503
nrCommits		1.000	0.704	0.996	0.773	0.748	0.732	0.466
nrAuthors			1.000	0.683	0.981	0.914	0.944	0.830
Power				1.000	0.756	0.732	0.714	0.439
dPower					1.000	0.943	0.964	0.772
ndCloseness						1.000	0.990	0.738
ndReach							1.000	0.773
ndBetweenness								1.000

Section 4.6 presents threats to validity of our experiments.

4.1 Correlation Analysis

For analyzing the correlation between network centrality measures and the number of post-release failures we determinedthe Spearman rank correlation values. We selected Spearman over Pearson correlation because Spearman is a more robust technique that can be applied even when the association between the measures is non-linear [11]. Spearman obtains correlation values between -1 and +1. -1 means high negative correlation, +1 high positive correlation, and 0 means that there is no correlation between two measures. We took values lower than -0.5 and larger than +0.5 denote significant correlation between two measures. Values lower than -0.7 and larger than +0.7 denote strong correlation.

The results of the Spearman correlation are shown in Table 1. As indicated by the high correlation values in the first row of Table 1 there are strong correlations between the network centrality measures and the number of post-release failures (*nrFailures*). Except the *ndBetweenness* measure all centrality measures exhibit a strong correlation with *nrFailures*. This result adds to the acceptance of hypothesis H1—network centrality measures are significant indicators for failure-prone binaries; and H2—the centrality of binaries in the contribution network correlates positively with the number of post-release failures.

We observe slightly higher correlation of up to 4.7% for advanced network centrality measures than for *nrAuthors* and *nrCommits*. This presents only a weak indicator to accept H3—advanced network centrality measures significantly improve failure prediction models. A more detailed discussion is needed and presented in Section 4.4.

Further details about the strong correlations between the network centrality measures and the number of post-release failures are presented in the next section.

4.2 Predicting Failure-Prone Binaries

We used binary logistic regression analysis for the evaluation of hypothesis H1—network centrality measures are significant indicators for failure-prone binaries. Logistic regression models compute the likelihood whether a binary is *failure-prone* or *not failure-prone* as a value between 0 and 1.

$$f\_prone(binary) = \begin{cases} 0, not\ failure-prone \\ 1, \quad failure-prone \end{cases}$$

We used a standard technique for evaluating the performance of binary logistic regression models, namely data splitting. 50 experiments were performed. In each experiment two third of the binaries (i.e., from the sample of 3500+ binaries) were randomly selected to train a model that then was applied to the other third of the binaries to classify them into *failure-prone* and *not failure-prone* binaries [30], [34]. We selected 0.5 as cut-off point so that a predicted probability less than 0.5 denotes a *not failure-prone* binary while a probability greater equal 0.5 denotes a *failure-prone* binary.

To compare and evaluate the performance of our models we use precision, recall, and AUC (area under the ROC curve). Precision and recall were computed based on the following classification table:

		predicted	
		not failure-p.	failure-prone
observed	not failure-p.	TN	FP
	failure-prone	FN	TP

Precision (P) denotes the proportion of correctly predicted failure-prone binaries: $P = TP/(TP + FP)$.

Recall (R) denotes the proportion of true positives of all failure-prone binaries: $R = TP/(TP + FN)$.

An ideal model has a precision and recall equal 1.0 which means all binaries were classified correctly. High recall is preferred over high precision because it is more cost effective to invest additional work in a likely failure-prone binary in advance than fixing possible failures after a release.

The receiver operating characteristic (ROC) is a technique to compare two operating characteristics namely the fraction of true positives with the fraction of false positives as the criterion changes [16]. The predicted probability is selected as the criterion for our experiments with binary logistic regression models. AUC represents a summary statistic of an ROC curve which can be interpreted as the probability, that, when randomly selecting a positive and a negative example the model assigns a higher score to the positive example. The higher the probability the better is the performance of a model.

Results of the correlation analysis in Table 1 show high multi-collinearity between the network centrality measures. High multi-collinearity between independent variables leads to high standard errors of the parameter estimates (i.e., b coefficients). This causes interpretations of the relative importance of an independent variable in the regression to be unreliable. We used Principal Component Analysis (PCA) to overcome multi-collinearity [12], [22]. PCA is used to reduce the dimensionality of a data set by

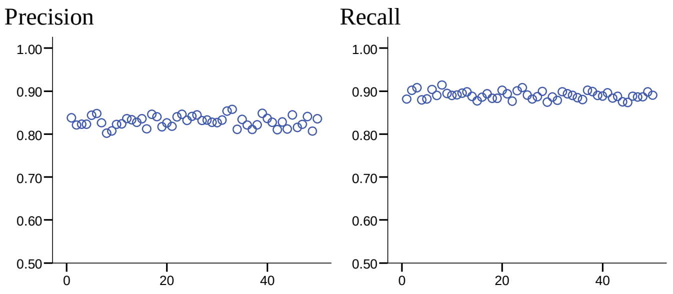


Figure 3. Performance measures of binary logistic regression with 50 random splits. Models are defined by four principal components that account for a cumulative variance greater than 95%.

retaining those characteristics that contribute most to its variance. For our experiments with binary logistic regression we used PCA to select uncorrelated linear combinations of independent variables which account for a cumulative variance greater than 95%.

Performing PCA with the set of centrality measures we obtained four principal components. These components then were used as independent variables in the binary logistic regression to classify binaries into failure-prone and not failure-prone.

The results of the 50 random splits are depicted by Figure 3. The following table lists descriptive statistics of presented scatter plots.

	Min	Max	Mean	Std. Dev.
Precision	0.802	0.858	0.829	0.013
Recall	0.874	0.914	0.890	0.009

Results show that network centrality measures present significant predictors for failure-prone binaries. In detail, failure-prone binaries can be predicted with an average precision of 0.829 and average recall of 0.89. This is significantly higher compared to the prior of 0.6. High ROC AUC values indicate high performance of computed regression models. Furthermore, scatter plots and low values for the standard deviation of performance measures are consistent over the 50 experiments indicating consistent models. Based on these results we accept hypothesis H1—network centrality measures are significant indicators for failure-prone binaries.

4.3 Predicting the Number of Post-Release Failures

Multiple linear regression analysis was used to investigate hypothesis H2. Our main objective of this experiment was to find out the network centrality measures that are most significant for predicting the number of post-release failures. Network centrality measures represent the independent variables and number of post-release failures (*nrFailures*) is the dependent variable.

In the previous experiment with binary logistic regression we showed that network centrality measures are multi- correlated. The use of PCA solved the problem of multi-collinearity, however, to the cost of readability and interpretability of resulting models [12]. Instead of using principal components we standardized the values of network centrality measures and used tolerance, variance-inflation factor (VIF), and collinearity diagnostics to deal with multi-collinearity. Tolerance for an independent variable is $1-R^2$ for the regression of that independent variable on all the other independent variables. An independent variable with a tolerance value close to 0 indicates that most of its variance is explained by the other independent variables, hence there is multi-collinearty. VIF is the reciprocal of tolerance consequently high values for VIF indicate multi-collinearity between independent variables. We used 0.2 as threshold for tolerance and 4.0 as threshold for VIF as proposed by the article on multiple regression at Statnotes.[2] In addition, we used the collinearity diagnostic of SPSS with eigenvalue and conditioning index. Eigenvalues close to 0 and condition indices greater 15 indicate highly intercorrelated predictors.

We further used scatter plots and P-P plots of standardized residuals to check linearity of linear regression models and normality of residual errors.

We started the linear regression analysis with all independent variables and inspected the model statistics and diagnostic results output by SPSS. We removed highly correlated independent variables from the model and re-computed the linear regression until the statistics showed no evidence for multi-collinearity between independent variables. With this method we retrieved the following linear regression model:

$$nrFailures = b_1 * ndCloseness + b_2 * nrAuthors + b_3 * nrCommits + a_0$$

b_i represent the coefficients and a_0 the constant of the model. According to this model the number of post-release failures of a Vista binary can be estimated by its closeness centrality (*ndCloseness*), the number of developers (*nrAuthors*), and the number of commits (*nrCommits*). Similar results were retrieved with models that used *ndReach* instead of *ndCloseness*. This is due to the high correlation between these two measures (see Table 1). Scatter-plots and P-P plots did not show any indicator of violating the linearity and normality assumptions.

We used data splitting for evaluating the performance of the linear regression models. As in the previous experiment with binary logistic regression, two third of the binaries were randomly selected to train a model. The model then was applied to the other third of binaries to predict their number of post-release failures. R-Square, Pearson and Spearman rank correlations were computed to measure the performance of each model. R-Square, also the coefficient of determination, is the percentage of the variance in the dependent variable explained by the regression model. Values range from 0 to 1 whereas a high R-Square value denotes a model with high statistical explanative power. Additionally, F-tests were performed on the regression models. They measure the statistical significance of a model based on the null hypothesis (i.e., regression coefficients are zero).

The results of the 50 experiments are depicted by Figure 4. Descriptive statistics for the measures of the scatter plots are listed in the table below.

	Min	Max	Mean	Std. Dev.
R-Square	0.698	0.746	0.716	0.010
Pearson	0.808	0.868	N/A	N/A
Spearman	0.752	0.805	N/A	N/A

The F-tests showed that the 50 models and correlations were significant at the 0.01 level. Scatter plots and low values for the

[2] http://www2.chass.ncsu.edu/garson/PA765/statnote.htm

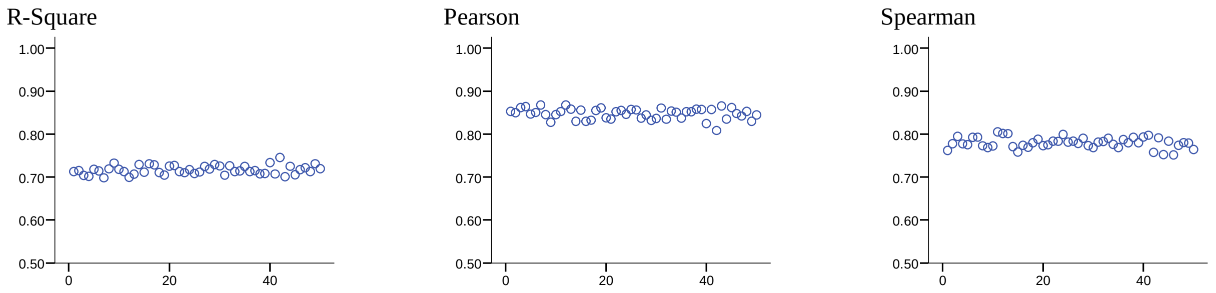


Figure 4. Performance measures of linear regression with 50 random splits. Models are defined by closeness centrality (*ndCloseness*), number of developers (*nrAuthors*), and the number of commits (*nrCommits*).

standard deviation of performance measures show consistent results. The mean value of R-Square is greater than 0.7 which denotes models with high statistical explanative power. This argument is further supported by consistent high values of the Pearson and Spearman correlations. Furthermore, Spearman rank correlations of around 0.78 demonstrate the sensitivity of linear regression models: an increase/decrease in the estimated values is accompanied by a corresponding increase/decrease of observed number of failures. An increase in *ndCloseness*, *nrAuthors*, and *nrCommits* values leads to an increase in the number of post-release failures. In other words, the more central a binary is in the contribution network and the more developers contribute to it the more failures the binary will have. The results of the linear regression analysis let us accept hypothesis H2—the more central a binary the more post-release failures it will have.

4.4 Significance of Advanced Centrality Measures

Addressing hypothesis H3 we investigated performance improvements of regression models due to the inclusion of advanced network centrality measures. For this we compared the performance of models with basic centrality measures with models that include advanced centrality measures. The set of basic network centrality measures *nrAuthors* and *nrCommits* is referred to as *BASIC*. Adding the advanced network centrality measures we get a set of measures that we refer to as *ADVANCED*.

Predicting failure-prone binaries with binary logistic regression: We first compared the significance of basic and advanced network centrality measures for predicting failure-prone Vista binaries. As before, we used binary logistic regression analysis. PCA was used to handle multi-collinearity of independent variables. For each set of measures we performed 50 random splits and compared the model performance statistics.

Figure 5 shows the results of the binary logistic regression analysis with *BASIC* measures on the left and *ADVANCED* measures on the right. The scatter-plots of both sets demonstrate consistent results over the 50 random splits. Comparing the mean values of performance measures, *BASIC* derives models with higher precision (0.869 compared to 0.829) at a cost of a slightly lower recall (0.877 compared to 0.89). Based on these results the advanced centrality measures *did not* significantly improve the performance of binary logistic regression models for predicting failure-prone binaries.

Predicting the number of post-release failures with linear regression: In the next experiment, we investigated the significance of advanced network centrality measures for predicting the number of post-release failures. We standardized network centrality measures to enable the assessment of the relative importance of independent variables in linear regression models. Figure 6 shows the results of the 50 random splits for models with *BASIC* measures and models with *ADVANCED* measures. Models and correlations of both measurement sets were significant at the 0.01 level.

All six scatter plots show consistent results over the 50 random splits. Comparing the R-Square values, the models obtained from the *ADVANCED* measures significantly outperform models obtained from *BASIC* measures. The mean of *ADVANCED* R-Square is 0.716 which on average is an increase by 0.326 over *BASIC* R-Square values. The plots for the Pearson correlation confirm this result.

Next, we investigated the significance of the advanced network centrality measure *ndCloseness*. In the last of the 50 models the coefficient of *ndCloseness* is 1.7X times higher than the coefficient of *nrAuthors* and 2.97Y times higher than the coefficient of *nrCommits*. An inspection of the other 49 regression models yielded similar results. The difference in coefficients clearly points out the relative importance of *ndCloseness* in our linear regression models. In particular, by comparing the models with the *BASIC* and *ADVANCED* measures we found out that on average the prediction of the number of post-release failures of Vista binaries can be improved by 32% when including the closeness centrality measure *ndCloseness*.

Combining the results from the binary logistic with linear regression analysis we partially accept hypothesis H3—advanced network centrality measures significantly improve prediction models.

4.5 Summary of Results

The results of the empirically study can be summarized as follows:

Network centrality measures can predict failure-prone Vista binaries. Our prediction models show similar performance as the models with organizational measures presented in [34] but significantly better performance than models with code churn and code complexity measures (see also [34]). Compared to [43] our models show a significantly higher recall and precision. The average precision of our models is 0.829 and average recall is 0.89. The two most significant predictors for failure-prone binaries are number of developers *nrAuthors* and number of commits *nrCommits*. Advanced network centrality measures *did not* significantly improve binary logistic regression models for

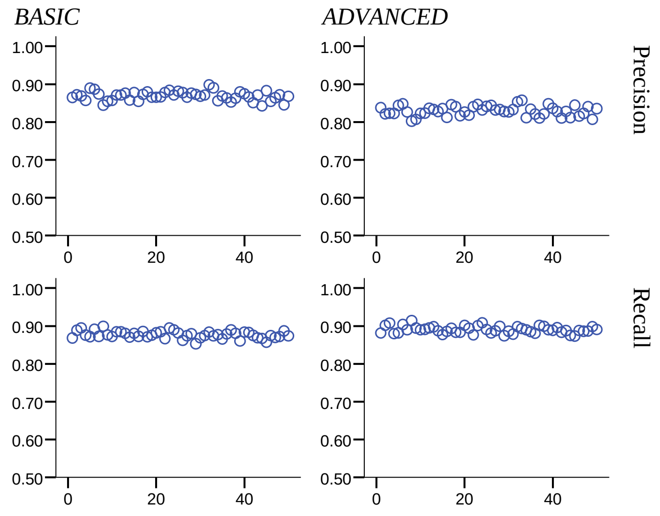


Figure 5. Comparison of performance measures of binary logistic regression models with *BASIC* and *ADVANCED* network centrality measures.

predicting failure-prone binaries. We recommend to use *nrAuthors* and *nrCommits* for predicting failure-prone binaries.

We accepted hypothesis H1—network centrality measures are significant indicators for failure-prone binaries.

Network centrality measures can predict the number of post-release failures. Average R-Square value of the 50 models is 0.716 which denotes models with high explanative power. This is a significant improvement when compared to the models that we obtained from static dependency graphs [43]. The most significant predictors are closeness centrality *ndCloseness*, number of developers *nrAuthors*, and number of commits *nrCommits*. Strong positive correlation of around 0.78 shows that an increase/decrease in predictor values leads to an increase/decrease in the number of post-release failures. We accepted hypothesis H2—the more central a binary the more post-release failures it will have.

Advanced network centrality measures can improve the prediction of number of post-release failures. Including the *ndCloseness* measure the R-Square value of linear regression models was improved by 0.326 and the prediction of the number of post-release failures of Vista binaries can be improved by 32% on average. The closeness centrality measure *ndCloseness* is the most significant predictor. Models with *ndReach* instead of *ndCloseness* showed similar performance. The advantage of the *ndReach* measure is its lower computational effort which is $O(N^2)$ compared to $O(N^3)$ for *ndCloseness*. Therefore, we recommend to use *ndReach*, *nrAuthors*, and *nrCommits* to predict the number of post-release failures. Combining the results of logistic and linear regression analysis, we partially accepted hypothesis H3—advanced network centrality measures significantly improve prediction models.

Application of results. We empirically validated that developer contributions need to be focused to reduce the number of failures. As a result, we can provide the Vista development team with a list of failure-prone binaries whose contribution structure needs to be adjusted to re-focus developer contributions. In addition, decisions can aim at a redesign of failure-prone binaries that provide diverse functionalities developed by different teams.

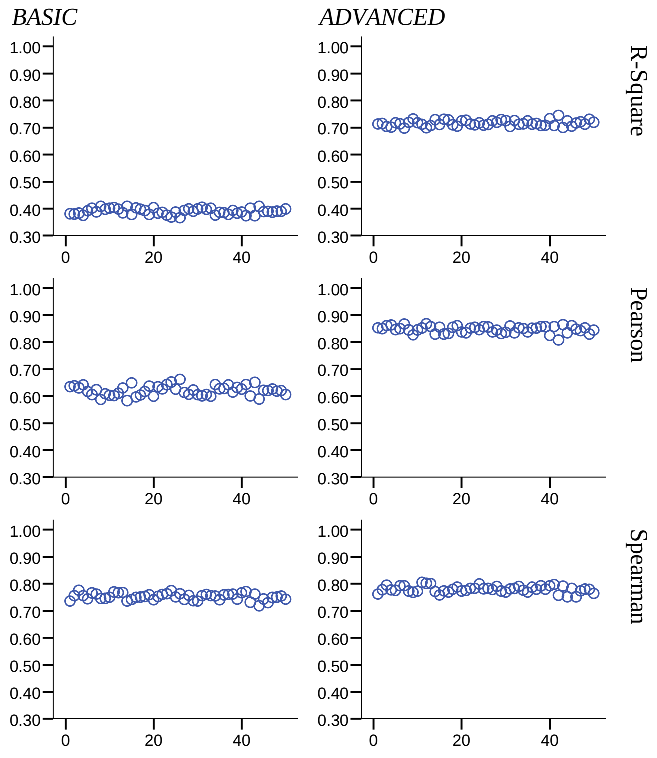


Figure 6. Comparison of performance measures of linear regression models with *BASIC* and *ADVANCED* network centrality measures.

Software development teams can benefit from our results. A careful discussion, however, is mandatory. The room for interpretation is large because the fragmentation of developer contributions is only another piece in the puzzle of why software systems fail. Organizational, process, and product measures need to be considered, as well, to provide a more complete picture. Furthermore, the following threats to validity of our experiments have to be taken into account.

4.6 Threats to Validity

In this section we discuss the main threats to validity of our experiment. From an external validity viewpoint, our experiment was performed using data collected from only one (though large) system, namely Windows Vista. It is possible that the results obtained in our studies may vary for different systems. We cannot assume a priori that the results of our study generalize beyond the specific environment in which it was conducted. Researchers become more confident in a theory when similar findings emerge in different contexts [2]. To alleviate this problem we are beginning to collect data from other Microsoft and open source systems to replicate these studies to build an empirical body of knowledge. The construct validity issues regarding the actual measurement of data being faulty are addressed to a large degree by the fact that the entire data collection and metrics computation process was completely automated by tools used in Microsoft for actual software development. The construct validity issues also arise from too few developers making edits to all the binaries, as might be the case with people who work in the build labs, people who instrument the binaries, collect code coverage data, are involved in the version control system maintenance, etc. From this purpose we defined the horizontal engineers (as discussed earlier in the paper) and removed them from the analysis so that their presence does not influence the results.

The internal validity issues for our experiment arise primarily from two sources. The first is 'researcher bias' which is alleviated by the fact that the authors worked in Microsoft Research, a parallel research arm of Microsoft with no intersecting management to the Windows team. Hence there was no motivation to show either way that binaries with fragmented contribution have less/more problems than binaries developed with focused contributions. The second threat is that we do not use the severity of a bug in the failure-prediction analysis. The reason for not using the severity stems from the fact that the severity numbers follow a sliding rule principle. A class of bugs maybe assigned the highest severity during the early stages of development but during the later stages might get a reduced severity. Hence this causes us to use calendar time, a measure not very accurate for use in analysis. In future studies we plan to incorporate the severity number by taking into account the exact time at which the fix was made.

5. RELATED WORK

In this section we present related work in the field of social network analysis in software engineering and defect/failure prediction.

5.1 Social Network Analysis

Social network analysis has been frequently used to study process and organizational aspects of software engineering. We give a brief summary of related approaches that applied social network analysis techniques to networks obtained from change log data.

Ghosh showed that many open source projects hosted at SourceForge.net are organized as self-organizing social networks [14]. Similarly, Xu et al. studied the development community at SourceForge.net and classified contributors into project leader, core developer, co-developer, and active user [42]. Huang et al. used historical data to identify core and peripheral development teams [20]. Ohira et al. used social networks and collaborative filtering to support the identification of experts across projects [35]. Lopez et al. explored statistics and social network properties of the development community at SourceForge.net to find collaborations and topological properties [26]. They found small world phenomenon and scale free behaviors and also that weakly associated but contributing co-developers and active users may be important factors in open source software development.

Howison et al. also used the SourceForge.net data to look at the dynamics of social structures by undertaking social network analysis over time [19]. They wanted to better understand how the social structures in projects are changing. Results showed that most of the participants are involved for only a short period while few participants are involved for long periods. Ducheneaut analyzed the socialization of newcomers to the OSS community of Python [10]. He combined the social network built from the mailing list archive with CVS log data to look at the trajectories of participants from joining to contributing. To visualize the project's evolution he implemented the OSS Browser, which provides a dynamic view of the social network, built on the Conversation Map of Sack [38]. Sack et al. performed an analysis across the three information spaces which build the socio-technical network: discussion, implementation, and documentation [39]. They tried to answer the questions how power is distributed, how links evolve between people, and how the cognitive activity of discussions is influenced by the social and governance structures of the project. Similarly, Bird et al. presented a study in which they analyzed the process by which people join open source projects [4]. Results provided support for their hypotheses that the rate of immigration is non-monotonic, and that technical skill and social reputation has an impact on becoming a developer. These approaches use social network analysis to investigate organizational, coordination, and communication aspects of software development. We use it to measure the centrality of software modules (binaries) in a contribution network and analyze its relationships to software quality.

5.2 Defect and Failure Prediction

In the past years researchers worked on finding out the most suitable set of measures and techniques to predict defects or failures in software modules and source files. While earlier work favored product measures obtained from source code recent research focused on process measures obtained from software repositories. Some approaches that investigated product measures for predicting defects/failures, in particular source code size and complexity measures, are presented in [1], [9], [12], [27], and [32]. Recently, Zimmermann and Nagappan presented a study with the Microsoft Windows 2003 server project [43]. They applied network centrality measures to the static dependency graph of Windows Server 2003 binaries to predict the probability and number of post-release failures. They were able to predict 60% of the escrow (or important) binaries—twice as many as identified with traditional program complexity measures. We use a similar set of network centrality measures but on the contribution network of the Microsoft Windows Vista project.

The predictive power of process measures has been confirmed by a number of empirical studies. To the best of our knowledge none of the related approaches used a contribution network and explored the relationships between the centrality of software modules and number of post-release failures. For example, Graves et al. explored the extent to which measurements from the change history are successful in predicting the number of faults in software modules [15]. They found out that process measures and in particular number of changes and the age of modules outperform product measures such as lines of code. Their best performing model was the weighted time damp model that weights changes according to their age (old changes are down-weighted). Knab et al. [24] also investigated various product and process measures obtained from the CVS and Bugzilla repositories of the Mozilla open source project to predict the number of failures in source files. Their results confirmed that process measures are significant predictors. Ratzinger et al. showed that process measures can also be used to predict short-term failures [36]. They used 63 product and process measures computed of two months of development time before a release to predict the number of failures that were reported in the following two months after the release. Bernstein et al. used process measures and non-linear models to predict failure-prone source files [3]. Results of their experiments with the Eclipse project showed that temporal features are significant predictors for failure-prone source files. This confirms the results of previous studies, such as Graves et al. [15]. The use of non-linear models turned out to be useful because of non-linear relationships between process measures. Recently, Moser et al. performed a comparative analysis of the efficiency of change metrics and static code attributes for defect prediction [29]. Three common machine learners were used to create prediction models with data obtained from the Eclipse project. Results clearly showed that process measures are more efficient defect predictors than product measures. Khoshgoftaar et al. [25] used process measures and step-wise binary logistic regression to predict whether a module is fault-prone at the end of the integration phase. In an empirical study with the JStar system they provide evidence that: 1) modules that had faults in the past are likely to have faults in the future; 2) unplanned requirement changes result in faults; 3) faults are more likely when code is changed. In their prediction approach with cached history Kim et al. elicited the evidence that changes lead to faults [23]. The cache keeps track of locations that are likely to have faults including any locations changed together with the fault, recently added locations, and recently changed locations. In experiments with seven open source projects they were able to predict faults at the file level with an accuracy of 73%-95% and 46%-72% at the entity level. These approaches empirically validated that process measures and in particular the number and location of past changes are significant predictors. In extension to these approaches we explore the centrality of software modules in a contribution network.

Mockus and Weiss developed a model to predict the risk of software changes [28]. They computed several measures of changes involved in an initial modification request (IMR). Binary logistic regression analysis was used to compute the probability with which an IMR will cause a failure. Based on the probability appropriate decisions regarding inspection, testing, and delivery can be made. In [18] they used data about software changes and survey data to model the extent of delay in a distributed development organization. A key finding is that the number of people involved is strongly related to the calendar time to complete an IMR. Ostrand et al. used a negative binomial regression model computed of product and process measures to predict the number of faults of source files in the next release [36]. While product measures were computed of the current release process measures were computed of the previous releases of source files. Results showed that 20 percent of the files with the highest predicted number of faults contained between 71% and 92% (83% on average) of the faults that were actually detected. In [41] they extended their prediction models to include developer information. This slightly improved their prediction model from 83.9% to 84.9%. While these approaches take into account the experience level of developers they do not take into account the extent to which developers concentrate their contributions on software modules.

Nagappan and Ball used code churn, which measures the amount of changes made to a component over a period of time, to predict system defect density [33]. Using statistical regression models on measures from the Windows Server 2003 project they found out that relative code churn measures can be used as efficient predictors of system defect density. In a follow-up of their work, Nagappan et al. recently presented a study on the statement by Brooks that product quality is strongly affected by organizational structure [34]. A set of measures to quantify organizational structures were obtained from the Microsoft Windows Vista project. Using binary logistic regression analysis they provide empirical evidence that organizational measures are significant predictors of failure-proneness. Our approach is motivated by the approach of Nagappan et al. and complements it. While they explored the impact of team fragmentation on software quality we explore the impact of fragmentation of developer contributions in general on software quality (without organizational structures).

6. CONCLUSION AND FUTURE WORK

In this paper we empirically investigated the relationship between the fragmentation of developer contributions and failure-proneness of software modules. We presented an approach to represent contribution structures with a contribution network and measure fragmentation of developer contributions with network centrality measures. In an empirically study with data from the Microsoft Windows Vista project we showed that the centrality of software modules in the contribution network can predict failures. In summary, the results of our study are:

- Network centrality measures *nrAuthors* and *nrCommits* can onm average predict more than 82.9% of *failure-prone* Vista binaries (see Section 4.2).
- Network centrality measures *ndCloseness*, *nrAuthors*, and *nrCommits* can predict the *number* of post-release failures of Vista binaries (see Section 4.3).
- Advanced network centrality measure *ndCloseness* or *ndReach* can improve the prediction of the *number* of post-release failures of Vista binaries by 32% (see Section 4.4).

Results support managers to inform organizational, process, and design decisions. For example, team and contribution structures of failure-prone binaries can be discussed and adjusted to re-focus developer contributions. In addition, decisions can aim at a redesign of failure-prone binaries that provide diverse functionalities developed by different teams.

Drawing general conclusions from empirical studies is difficult and we cannot assume that our findings can be generalized beyond the specific environment [2]. For this we plan to replicate the study with a number of open source and other Microsoft projects. Furthermore, fragmentation of developer contributions is only another piece in the puzzle of why software systems fail. We need to take into account organizational, process, and product measures to provide a sound basis for project decisions. Finally, it is the humans who introduce errors that later on lead to failures. In this direction, we plan collaborations with psychologists to add their perspective and factors that affect software quality.

7. ACKNOWLEDGMENTS

We would like to thank Harald C. Gall and Abraham Bernstein for their support and input on failure prediction models. Many thanks goes to Andrew Begel, and Tom Ball for discussions on developer contribution structures at Microsoft. We would also like to thank the anonymous FSE reviewers for their helpful comments.

8. REFERENCES

[1] V. R. Basili, L. C. Briand, and W. L. Melo. A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Transactions on Software Engineering*, 22(10), pp. 751-761, 1996.

[2] V. Basili, F. Shull, and F. Lanubile. Building Knowledge through Families of Experiments, *IEEE Transactions on Software Engineering*, 25(4), pp. 456-473, 1999.

[3] A. Bernstein, J. Ekanayake, and M. Pinzger. Improving defect prediction using temporal features and non linear models. In *Proceedings of the International Workshop on Principles of Software Evolution (IWPSE)*, pp. 11-18, IEEE CS Press, 2007.

[4] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and G. Hsu. Open Borders? Immigration in Open source Projects. In *Proceedings of the International Workshop on Mining Software Repositories (MSR)*, pp. 6, IEEE CS Press, 2007.

[5] P. Bonacich. Power and Centrality: A family of Measures. *American Journal of Sociology 92*, pp. 1170-1182, 1987.

[6] S. P. Borgatti, M. G. Everett, and L. C. Freeman. Ucinet for Windows: Software for Social Network Analysis. *Analytic Technologies*, Harvard, MA, 2002.

[7] A. Cockburn. Agile Software Development. Addison-Wesley, Boston, 2001.

[8] T. DeMarco and T. Lister. Peopleware: Productive Projects and Teams, 2nd edition, Dorset House Publishing, New York, 1999.

[9] N. E. Fenton and N. Ohlsson. Quantitative Analysis of Faults and Failures in a Complex Software System. *IEEE Transactions on Software Engineering*, 26(8), pp. 797-814, 2000.

[10] N. Ducheneaut. Socialization in an Open Source Software Community: A Socio-Technical Analysis. *Computer Supported Cooperative Work (CSCW)*, 14(4), pp. 323–368, 2005.

[11] N. E. Fenton and S. L. Pfleeger. Software Metrics: A Rigorous and Practical Approach. PWS Publishing Co., Boston, MA, USA, 1997.

[12] N. E. Fenton and M. Neil. A Critique of Software Defect Prediction Models. *IEEE Transactions on Software Engineering*, 25(5), pp. 675-689, 1999.

[13] L. C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1(3), pp. 215-239, 1979.

[14] R. A. Ghosh. Clustering and dependencies in free/open source software development: Methodology and tools. First Monday, 8(4), 2003.

[15] T. L. Graves, A. F. Karr, J. S. Marron, and H. Siy. Predicting Fault Incidence Using Software Change History. *IEEE Transactions on Software Engineering*, 26(7), pp. 653-661, 2000.

[16] D. M. Green and J. M. Swets. Signal detection theory and psychophysics. John Wiley & Sons, Inc.. New York, 1966.

[17] R. A. Hanneman and M. Riddle. Introduction to social network methods. University of California, Riverside, California, 2005.

[18] J. D. Herbsleb and A. Mockus. An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering*, 29(6), pp. 481-494, 2003.

[19] J. Howison, K. Inoue, and K. Crowston. Social Dynamics of Free and Open Source Team Communication. In *Proceedings of the International Conference on Open Source Software (OSS)*, pp. 319-330, 2006.

[20] S.-K. Huang and K.-m. Liu. Mining version histories to verify the learning process of Legitimate Peripheral Participants. In *Proceedings of the International Workshop on Mining Software Repositories (MSR)*, pp. 1-5, ACM Press, 2005.

[21] W. S. Humphrey. TSP: Leading a Development Team. Addison-Wesley, Boston, 2006.

[22] J. E. Jackson. A User's Guide to Principal Components. John Wiley & Sons, Inc., New York, 1991.

[23] S. Kim, T. Zimmermann, E. J. Whitehead, Jr., and A. Zeller. Predicting Faults from Cached History. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 489-498, IEEE CS Press, 2007.

[24] P. Knab, M. Pinzger, and A. Bernstein. Predicting defect densities in source code files with decision tree learners. In *Proceedings of the International Workshop on Mining Software Repositories (MSR)*, pp. 119-125, ACM Press, 2006.

[25] T. M. Khoshgoftaar, E. B. Allen, R. Halstead, G. P Trio, and R. M. Flass. Using Process History to Predict Software Quality. *IEEE Computer*, 31(4), pp. 66-72, 1998.

[26] L. Lopez-Fernandez, G. Robles, and J. M. Gonzalez-Barahona. Applying Social Network Analysis to the Information in CVS Repositories. In *Proceedings of the International Workshop on Mining Software Repositories (MSR)*, pp. 101-105, 2004.

[27] T. Menzies, J. Greenwald, and A. Frank. Data Mining Static Code Attributes to Learn Defect Predictors. *IEEE Transactions on Software Engineering*, 33(1), pp. 2-13, 2007.

[28] A. Mockus and D. M. Weiss: Predicting risk of software changes. *Bell Labs Technical Journal*, 5(2), pp. 169-180, 2000.

[29] R. Moser, W. Pedrycz, and G. Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 181-190, IEEE CS Press, 2008.

[30] J. C. Munson and T. M. Khosghoftaar. The Detection of Fault-Prone Programs. *IEEE Transactions on Software Engineering*, 18(5), pp. 423-433, 1992.

[31] J. C. Munson and S. G. Elbaum. Code Churn: A Measure for Estimating the Impact of Code Change. In *Proceedings of the International Conference on Software Maintenance (ICSM)*, pp. 24-32, IEEE CS Press, 1998.

[32] N. Nagappan, T. Ball, and A. Zeller. Mining metrics to predict component failures. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 452-461, ACM Press, 2006.

[33] N. Nagappan and T. Ball. Using Relative Code Churn Measures to Predict System Defect Density. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 284-292, ACM Press, 2005.

[34] N. Nagappan, B. Murphy, and V. Basili. The influence of organizational structure on software quality: an empirical case study. In Proceedings of the International Conference on Software Engineering (ICSE), pp. 521-530, IEEE CS Press, Leipzig, Germany, 2008.

[35] M. Ohira, N. Ohsugi, T. Ohoka, K. Matsumoto. Accelerating cross-project knowledge collaboration using collaborative filtering and social networks. In *Proceedings of the International Workshop on Mining Software Repositories (MSR)*, pp. 1-5, ACM Press, 2005.

[36] T. J. Ostrand, E. J. Weyuker, and R. M. Bell. Predicting the Location and Number of Faults in Large Software Systems. *IEEE Transactions on Software Engineering*, 31(4), pp. 340-355, 2005.

[37] J. Ratzinger, M. Pinzger, and H. C. Gall. EQ-Mine: Predicting short-term defects for software evolution. In *Proceedings of the Fundamental Approaches to Software Engineering (FASE)*, pp. 12-26, LNCS, Springer, 2007.

[38] W. Sack. Conversation Map: An Interface for Very Large-Scale Conversations. *Journal of Management Information Systems*, 17(3), pp. 73–92, 2001.

[39] W. Sack, F. Détienne, N. Ducheneaut, J.-M. Burkhardt, D. Mahendran, and F. Barcellini. A Methodological Framework for Socio-Cognitive Analysis of Collaborative Design of Open Source Software. *Computer Supported Cooperative Work (CSCW)*, 15(2), pp. 229–250, 2006.

[40] S. Wasserman and K. Faust. Social Network Analysis: Methods and Applications. Cambridge University Press, New York, 1994.

[41] E. J. Weyuker, T. J. Ostrand, and R. M. Bell. Using Developer Information as a Factor for Fault Prediction. In *Proceedings of the International Workshop on Predictor Models in Software Engineering (PROMISE)*, pp. 8-14, IEEE CS Press, 2007.

[42] J. Xu, Y. Gao, S. Christley, and G. Madey. A Topological Analysis of the Open Souce Software Development Community. In *Proceedings of the Annual Hawaii International Conference on System Sciences (HICSS)*, pp. 198, IEEE CS Press, 2005.

[43] T. Zimmermann and N. Nagappan. Predicting defects using network analysis on dependency graphs. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 531-540, IEEE CS Press, 2008.