Merits of Organizational Metrics in Defect Prediction: An Industrial Replication

Conference Paper · May 2015

DOI: 10.1109/ICSE.2015.138

CITATIONS
5

READS
81

6 authors, including:

Burak Turhan
Brunel University London
148 PUBLICATIONS **1,358** CITATIONS
SEE PROFILE

Ayse Bener
Ryerson University
177 PUBLICATIONS **1,799** CITATIONS
SEE PROFILE

Andriy V. Miranskyy
Ryerson University
49 PUBLICATIONS **119** CITATIONS
SEE PROFILE

Enzo Cialini
IBM
15 PUBLICATIONS **33** CITATIONS
SEE PROFILE

Merits of Organizational Metrics in Defect Prediction: An Industrial Replication

Bora Caglayan*, Burak Turhan†, Ayse Bener‡, Mayy Habayeb‡, Andriy Miransky§ and Enzo Cialini¶
*Department of Mathematics, Ryerson University, Canada
bora.caglayan@ryerson.ca
†Dept. of Information Processing Science, University of Oulu, Finland
Email: burak.turhan@oulu.fi
‡Mechanical and Industrial Engineering, Ryerson University, Canada
Email: {ayse.bener, mayy.habayeb}@ryerson.ca
§Department of Computer Science, Ryerson Unniversity, Canada
Email: avm@ryerson.ca
¶IBM Toronto Software Laboratory, Canada
Email: ecialini@ca.ibm.com

***Abstract*—Defect prediction models presented in the literature lack generalization unless the original study can be replicated using new datasets and in different organizational settings. Practitioners can also benefit from replicating studies in their own environment by gaining insights and comparing their findings with those reported. In this work, we replicated an earlier study in order to investigate the merits of organizational metrics in building defect prediction models for large-scale enterprise software. We mined the organizational, code complexity, code churn and pre-release bug metrics of that large scale software and built defect prediction models for each metric set. In the original study, organizational metrics were found to achieve the highest performance. In our case, models based on organizational metrics performed better than models based on churn metrics but were outperformed by pre-release metric models. Further, we verified four individual organisational metrics as indicators for defects. We conclude that the performance of different metric sets in building defect prediction models depends on the project's characteristics and the targeted prediction level. Our replication of earlier research enabled assessing the validity and limitations of organisational metrics in a different context.**

I. Introduction

Most of the research findings in the defect prediction literature are isolated in the papers published by a particular research group or laboratory. Open dataset repositories such as Promise make it easy to store and share datasets used in defect prediction [18]. In theory, if there is enough detail about the original study, the results can be validated independently by other researchers [8], [15]. However, dataset sharing may not be possible for some companies due to confidentiality concerns. Another problem is the ambiguous definitions in the research papers. For example, in most defect prediction papers, some information about the empirical setup and the data extraction process is missing [11].

Replication studies in software engineering (SE) are important to build a consistent body of knowledge. Overemphasis on novelty in software engineering research may slow down the progress [22]. Replication studies enable researchers and practitioners to disseminate novel approaches and methods into different contexts.

Replication of previous research is also valuable for the industry. The domain, scope and process related variables may be different for each project and organization. No solution provides a "silver bullet" that is applicable under all circumstances. Identifying the conditions under which a particular result is valid might help organizations to identify the research findings that are relevant to them.

In recent years, researchers have been investigating the merits of new metrics (i.e., product, people, process) in defect prediction. In this paper, we focused on the organizational metrics to explore: the relationship between organizational complexity and software quality, measures of the organizational structure, and their capability of predicting software quality in terms of defects. Organizational metrics are extracted from the organization structure of a software development unit. For each code module, based on the organizational hierarchy of the developers, these metrics capture the code ownership.

Our industry partner wanted to investigate the value of organization metrics in building defect prediction models. Estimation of the organizational metrics requires a well defined organigram for an organization with hierarchical teams. Our industry partner have a tree structure in terms of hierarchy, and in our study we used a set of architectural functionality in a large scale commercial software as our dataset. We used a highly cited paper as our baseline by Nagappan et al. from Microsoft Research [19]. In the original study the authors compared organizational metrics with other well known metric sets in building a defect prediction model. We used the same research approach, metrics and empirical design to the best possible extent. Then, we compared our findings with the original study to check the external validity of their findings.

The structure of this paper is based on the replication study framework suggested by Carver [8]. In the related work section, we give an overview the replication studies in empirical software engineering as well as an overview of organizational

metrics in defect prediction. In the description of the original study section, we overview the shared research question of this study (with the original study), context description, data collection, study design and the summary of the results of the original paper. In the information about the replication section, we describe our motivation, level of interaction with the original study researchers and the study changes. In the comparison of the results section, we compare our findings with the findings of the original study. In the threats to validity section, we overview the potential threats to the validity of the replication and conclude the paper with a brief summary of the findings and future directions.

II. Related Work

A. On the Value of Replication Studies in SE

In software engineering, replication studies are important to better understand different aspects of software development, to generalize from findings under different contexts, and to build best practice solutions and standards for the practice. Various research groups have emphasized the value of replication in the software engineering domain [13], [15], [16], [20], [25]. While Kitchenham et al. identifies replications as one of the key enablers to achieve an evidence based software engineering practice [16], Juristo et al. points to field replications using real data as a maturity parameter for replication studies [15].

Several studies have attempted to define and classify replication. Shull defined a replication to be a study that is run based on the design and results of a previous study [23]. He mentioned that the goal of replication is to either verify or broaden the applicability of the results of the initial study. Brooks et al. further categorized replications as either internal or external replications [6]. Internal replications are undertaken by the original researchers and they may be published as one paper or as a series, whereas, external replications are undertaken by independent researchers who seek to check and improve on the findings of other researchers. Sjoberg et al. investigated the amount of replicated studies and found that only one out of seven studies was replicated by an external party [24]. Further, Basili et al. have defined 6 types of replications [2]: 1) Strict replications, 2) Replications that vary variables intrinsic to the object of study, 3) Replications that vary variables intrinsic to the focus of the evaluation, 4) Replications that vary context variables in the environment in which the solution is evaluated, 5) Replications that vary the manner in which the study is run, 6) Replications that extend the theory.

In this respect, our study should be considered as a non-strict external replication in a different context (type #6). Empirical studies in software engineering are highly dependent on the environment, the organization of the development teams, the development process and the methodology. Therefore, external replications that take place in a different environment, if the same results are obtained, would lead towards better generalization of results.

In reporting our work, we used the replication *reporting guidelines* proposed by Carver [8]. Carver mainly suggests reporting the details of the original study, replication and the comparison of the findings in a replication paper. We chose to follow these guidelines since it allows researchers to report the similarities and differences between the original study and the replication. The reason for parallel and conflicting findings can also be observed using such a template.

B. Organizational Metrics in Defect Prediction

Organizational complexity can be defined within the context of interaction mechanisms among the components of an organization [10]. Organizational complexity and its effects on software quality has been previously addressed by several research groups. Conway identified the relationship between systems design and communication structures of the organizations, a.k.a. Conway's rule [10].

Programmer collaboration networks and the effect of programmer collaborations on software quality has also been investigated extensively. Herbsleb and Mockus investigated the relations between distributed software teams, increased software development costs and reduced productivity [12]. They also examined if works in different sites are interdependent and how such interdependence may diminish over time. They found that teams located at different sites have reduced overall efficiency by examining the software changes [12].

Similarly, Jimenez et al. [14], Cataldo and Nambiar [9], and Bird et al. [4] have all studied the effect of distributed software development efforts on software quality. Jimenez et al. surveyed the literature related to distributed systems development and summarized the challenges and improvements for distributed software development. Cataldo and Nambiar concluded that as software development work becomes more distributed the benefits of process improvement diminish. Another study by Bird et al. analyzed the post release failures for two systems, one developed in a distributed environment whilst the other was developed in a collocated environment, and they found negligible differences.

Caglayan et al. [7] studied the natural team formation in software projects by investigating the evolution of the collaboration network over time during a release of a large-scale project. They found that collaboration teams among the developers may be formed over time, independent of the formal team structure of the organization.

Another notable paper in this area is a recent paper by Bettenburg and Hassan [3] . They analyzed the issue level collaboration information in the Eclipse project. They checked the relation of the defect proneness of code with the collaboration and extracted several metrics to build a logistic regression model to predict the defect prone modules. They found that when a part of the software module is discussed in the issue management system by developers, the likelihood of a post release defect increases.

Programmer collaboration has been used previously to build metric sets for the defect prediction problem. Meneely et al. [17] and Alhassan et al. [1] tested the merits of the local collaboration metrics in the defect prediction problem. They found that the local collaboration metrics form a strong

alternative to the other well-established metric sets. Bird et al. [5] further investigated the ownership dimension in more detail, they extracted ownership metrics and evaluated the relationship of these metrics with pre-release and post release defects, and they concluded that there is a direct relationship between ownership metrics and post release defects. Organization structures have further been evaluated by Shihab et al. [21], focusing on software branching. They concluded that misalignment between the branching structure within software and the organizational structure is associated with higher post-release failure rates.

The most relevant work to ours — as a replication study — is the baseline, or the original study by Nagappan et al. which is explained in the next section in detail [19].

III. Description of The Original Study

In this section, we briefly provide information about the original study that we use as the basis of this replication. We follow Carver's 'replication guidelines' in reporting the recommended amount of details about the original study [8].

The motivation behind the original study, conducted by Nagappan et al., was to provide answers to the following questions [19]:

- How does organizational complexity influence quality?
- Can we identify measures of the organizational structure?
- How well do they do at predicting quality, e.g., do they do a better job of identifying problem components than previously used metrics?

To address these questions, Nagappan et al. investigated the "...relationship between organizational structure and software quality" [19] via a case study and analysis of data from Windows Vista. Their dataset included metric data (explained later on in this section) from 3404 binary files (corresponding to 50M+ lines of code) and their post-release failure information.

Nagappan et al. proposed a metric suite of eight measures to quantify organizational issues "...such as organizational distance of the developers; the number of developers working on a component; the amount of multi-tasking developers are doing across organizations; and the amount of change to a component within the context of that organization" [19]. The proposed metric suite[1] includes:

- **Number of Engineers (NOE)**: Number of unique people who edited the code base and who were working for the organization at the product release date.
- **Number of Ex-Engineers (NOEE)**: Number of unique people who edited the code base and who were not working for the organization anymore at the product release date.
- **Edit Frequency (EF)**: Total number of times the source code for the unit of analysis (i.e., binary files) is edited - measured by the number of commits in the source code revision system.
- **Depth of Master Ownership (DMO)**: The level of the person (e.g. Master Owner) in the organization tree, whose subtree (i.e., reporting engineers/ subordinates) contributes more than 75% of the changes.
- **Percentage of Organization contributing to development (PO)**: The ratio of the size of (e.g. number of people in) the subtree rooted at Master Owner to the total size of the (sub-) organization.
- **Level of Organizational Code Ownership (OCO)**: Percentage of changes made from the organization that contains the Master Owner.
- **Overall Organizational Ownership (OOW)**: The ratio of the size of (e.g. number of people in) the subtree rooted at Master Owner to the total number of people who made changes.
- **Organizational Intersection Factor (OIF)**: Number of different organizations who contribute changes that account more than 10% of all changes.

The original study validated the organizational metric suite by two different approaches. First, the authors constructed a step-wise regression model with backward selection. They observed that all eight metrics were retained in the final model. Second, they applied Principal Component Analysis (PCA). They observed that the number of principal components to explain the majority of the sample variance was eight, meaning that PCA cannot find less number of factors. Hence, they concluded "...that all the eight organizational measures contribute towards explaining the variance in the post-release failures..." and decided to use all eight measures in building models to predict failure-proneness [19].

The original study then built logistic regression models from organizational metric suite using random splits (2/3 for training and 1/3 for testing) of Windows Vista data. The decision threshold for the logistic regression was set to 0.5 for classifying an instance as defect-prone or not. The stability of the models across 50 random runs were reported in terms of precision, recall and the Spearman correlation between the predicted probability of fault-proneness and the dependent variable (i.e., post release defects in a binary file).

Besides the organizational metrics, the original study also included other, more traditional, metrics to be used as independent variables in constructing different conceptual models to predict failure-proneness of binary files in Windows Vista (i.e., whether a binary file has post-release defects). This resulted in a total of six comparable logistic regression models representing each metric dimension, i.e., organizational, code churn, code complexity, dependencies, code coverage and pre-release defects.

Table I provides (as per recommendations in [8]) a summary of the six comparable models as reported in the original study. The performance is assessed in terms of precision and recall, and the average of 50 random splits are reported in Table I. Nagappan et al. state that the performance achieved through the organizational structure model is significantly better than other models, with a 7.6% difference with the precision of the closest model corresponding to hundreds of binaries in

[1]Please refer to the original study for more detailed descriptions of the motivation and the theory behind these measures.

TABLE I: Summary of original study results (from [19]).

Model	Precision	Recall
Organizational Structure	86.2%	84.0%
Code Churn	78.6%	79.9%
Code Complexity	79.3%	66.0%
Dependencies	74.4%	69.9%
Code Covergae	83.8%	54.4%
Pre-Release Bugs	73.8%	62.9%

their context. They conclude "...that organizational metrics are better predictors of failure-proneness than the traditional metrics used so far" [19].

IV. Information About The Replication

In this section, we provide information about our motivation for conducting the replication, the level of interaction with the original authors and the changes to the original study.

A. Motivation for Conducting the Replication

Churn and complexity metrics are well-known metric sets in the defect prediction literature [11]. On the other hand, organizational metrics is a new metric set proposed by the authors of the original study. These metrics model the relation between the organizational structure and the source code. In the original study by Nagappan et al. they built a prediction model that used organizational metrics. This model outperformed previous models using several well known product related metric sets as well as churn metrics [19] in Windows Vista. Based on these findings and considering their ease of extraction, they concluded that organizational metrics seems to be a promising new metric set. Although this paper has been widely cited, we have not found any other study that used these metrics on a different setting and dataset. The purpose of our study is to test the merits of organizational metrics on another large scale commercial software. We performed a close replication of the original study on a dataset from our industrial partner.

B. Level of Interaction with the Original Researchers

In a typical software engineering research paper, it is not possible to explain every detail of the study. Therefore, the level of interaction with the original researchers is important to clarify these details. In this study we had personal communications with the authors of the original paper to clarify methodological issues. Nevertheless, operating in a different industrial context and based on our industry partner's interests, our study included some changes in the study context, which are explained next.

C. Changes to the Original Study

It is important to note that our methodology follows the methodology in the original paper as much as possible. In this section, we report the deviations from the original study due to changes in the operational context and our industry partner's interests.

Naturally, the most important change in our empirical setup is related to dataset. The main difference of our study is the targeted level of defect prediction. We used C/C++ source code functions in our case instead of binaries as input for the defect prediction model. All of the metric calculations were done on source code function level. In our study, we had to do a few changes on data extraction due to the differences in the project that we analyzed. The rest of this section explains these differences in a more detailed manner.

1) Different Dataset: We used a large-scale enterprise software product to conduct our empirical work. The enterprise software product of the company has a 20 years old code base. We used a 1467 kLOC part of the product as the dataset. The software is a set of architectural functionality and a core part of a large relational database management software with 30 MLOC. Contextual details of the dataset are included as part of Table II in comparison with the original study.

In the original study, authors predicted defects in binary file level. Binaries are a compiled set of source code files in Microsoft Vista. In our study, although we had the option to follow the original study, due to company goals of defect management, our industry partner wanted us to operate at a much lower level. In our case, quality (defect) info is directly available at file level and it provides more focused ability to take action. Mean LOC per file was more than 3 kLOC and a typical file in our dataset contained more than 500 functions. Due to large file size, nearly every file had at least one defect during the observed period. The total size of the dataset used was 1.5m LOC and 4855 functions. Compared to the size of Windows Vista (3404 binaries, 50m LOC) our project was considerably smaller in terms of LOC.

The other difference is about the purpose and the users of projects. Microsoft Windows is a popular operating system with hundreds of millions of users. On the other hand, the project we used in this study is a set of functionality used by other components of the relational database management system and are accessed only indirectly by the end-users.

In the original study, the details of the developer hierarchy in the Windows Vista team are not provided. The authors advised using organizational metrics for projects with more than thirty developers. In our case, hierarchy for most teams is four level deep and the number of developers[2] is well over 30. A part of the developer reporting hierarchy in the organization is provided as a figure in the appendix.

2) Missing Metric Sets: We were able to extract metrics at four dimensions as defined in the original study, namely: static code, churn, pre-release defect and organizational metrics. We could not extract dependency and code coverage data since these metrics were not stored by the organization.

We used all of the original organizational metrics in the project except number of ex-engineers since no developer left the project during the observed period. The most important part of organizational metric extraction process is building an accurate representation of the organization's organigram.

[2]Due to confidentiality issues we are not able to report the total number of developers, but it is sufficient to say that it is 30+ in a 4+ level depth hierarchy, which is more than the recommended minimum by the original study.

TABLE II: Data Set of Original Study and Dataset of Replication Study

	Original	Replication
Project	Windows Vista	A Set of Architectural Functionality in an Enterprise Software
Programming Language	C and C++	C and C++
Versioning System	NA	IBM Rational ClearCase
Software Module Definition	Binary	Function (in C/C++ sense)
Number of Modules	3404	4855
Size (LOC)	50 Million	1.5 Million
Snapshot Date	NA	10 months before release

We initially extracted this organigram with a bottom up approach. For each issue in the project, manager and developer information was stored. We formed a tree by merging all these relations. Initially the merging procedure produced a lot of disconnected trees. We held several meetings with our contacts in the organization to merge these trees and produce an accurate organigram for the project.

We could not compare our metric extraction methodology with the one employed in the original paper. We believe, these details had to be omitted by the authors because of the privacy concerns in the company. The authors may have had access to the actual organigram in Microsoft considering their affiliations with the company.

3) Differences in Defect Prediction Target Entities: As explained in the dataset section, the most important difference in our empirical setup is the definition of the targeted software modules for the sake of defect prediction. Source code function (in C, C++ sense) level is much finer grained compared to binary level since a binary may consist of several source code files. In contrast, a C source code file may have tens or hundreds of functions. We used the same performance measures for estimating the defect prediction performance. Similarly, we used the same study design as the original study. The dataset were divided into 1/3 and 2/3 random splits and the analyses were repeated 50 times. Distribution and the means of Spearman correlation, precision and recall values were reported. The study design in the original paper can be seen in Section III.

V. Comparison of The Results

In this section, we report our findings, and the differences with the original study are denoted in *italic face*.

A. Exploratory Analyses

In the first phase, we checked the distributions of the organizational metrics. In Figure 1, the pairwise distributions of the organizational metrics can be observed. Similarly, Table III shows the correlations among the organizational metrics. The correlation data or the pairwise distributions were not provided in the original study. In our case, the correlation among the organizational metrics are high. In the figure and the table, we see very high positive correlation between number of engineers (NOE) and edit frequency (EF). This finding can be intuitively expected since the number of edits would increase with the increasing number of unique developers who change a piece of code. One might also argue that, EF and NOE metrics are related to churn rather than organization. The other five organizational metrics had relatively lower correlation among themselves, except for OOW with NOE, OCO and OIF. Since OOW is defined in terms of the ratio of the Master Owner (root of subtree that contributes 75% of changes), its strong correlations with NOE and OCO do not come as a surprise.

TABLE III: Correlation among the organizational metrics. ((Significance: $*: P < 0.05, **: P < 0.01, ***: P < 0.001$)

	noe	ef	oco	oif	po	oow
noe						
ef	0.84***					
oco	-0.57***	-0.27***				
oif	0.54***	0.26***	-0.73***			
po	-0.05***	-0.03*	0.06***	-0.07***		
oow	-0.60***	-0.33***	0.62***	-0.70***	0.08***	
dmo	0.00	0.02	0.06***	-0.02	0.00	0.03*

We then performed Principal Component Analysis (PCA) on the organizational metrics to check principal components (PC's). The authors summarized their PCA results as follows: "Running a PCA on the eight organizational measures resulted in the generation of eight principal components indicating that PCA does not reduce the computation overhead in anyway by transforming the organization measures into fewer factors which can be used as predictors" [19]. However, PCA always generates as many PC's as the number of features, and in the original study raw numbers for the proportion of variance explained have not been provided.

In our case, when we performed PCA on the normalized metrics the results were *different*. The first five components explained more than 95 percent of the variance (see Table IV). Although in general terms we agree with the authors' remark about the computational overhead, considering the total number of features, the computational overhead caused by increased number of features is negligible.

In summary, we have observed some differences in validating the individual organizational metrics, though we followed

TABLE IV: Proportion of Variance Explained by The PCA Components

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7
Standard deviation	1.79	1.05	1	0.98	0.62	0.49	0.29
Proportion of Variance	0.46	0.16	0.14	0.15	0.05	0.03	0.01
Cumulative Proportion	0.46	0.62	0.76	0.90	0.95	0.99	1

TABLE V: Values of the coefficients for the attributes. $F-Stat$ criterion was used to remove attributes. (Significance: $. : p < 0.1, * : P < 0.05$).

	Df	Deviance	F value	Pr(>F)	Sign.
<none>		6254.21			
noe	1	6300.21	35.65	0.0000	*
ef	1	6258.55	3.37	0.0666	.
oco	1	6255.33	0.87	0.3500	
oif	1	6255.23	0.79	0.3738	
po	1	6366.59	87.08	0.0000	*
oow	1	6261.90	5.96	0.0147	*
dmo	1	6254.34	0.10	0.7472	

the analyses procedure of the original study. This is a sign that the effectiveness or descriptive power of the proposed metrics may depend on the context of the project/ organization in which they are applied.

B. Comparison of Prediction Model Outputs

In this step, we compared the defect prediction performance of the organizational metrics in our dataset with the results of the original study.

The authors of the original study found all of the organizational metrics to be significant in the multivariate model, e.g. all of them were retained in the logistic regression model with backward selection. [19]. The coefficients for these metrics were not provided in their paper. In Table V the coefficients and the significance values for the organizational metrics for the model based on our dataset are provided. The non-significant attributes were dropped in the final stepwise-regression model that generated the results discussed in this section.

In our case, only four out of eight metrics, namely the number of editors (NOE), edit frequency (EF), percentage of organization contributing to development (PO) and overall organizational ownership (OOW) were found to be significant.

For convenience, we show the stability indicators for the results of the original study and the replication side by side for easy comparison, in Figures 2, 3 and 4 (used in the original study and reproduced with our dataset). These figures show the fluctuations in recall, precision and Spearman correlation (among the predictions and actual defect info for target modules) performance metrics for 50 randomized runs. The fluctuations in the performance for the randomized runs seem to be similar in both cases, indeed mostly stable, based on visual inspection.

However, our performance indicators seem to be consistently lower than the original results, especially for Spearman correlations. This difference in the predictive power of organisational metrics can again be explained by the different context of our dataset.

Finally, the authors also reported the mean of precision and recall values in the original paper. The comparison of the means for the two datasets can be seen in Table VI. We checked the significance of the differences in the performance four conceptual models based on different metric sets (i.e., organisational structure, code churn, code complexity, pre-release bugs) using Mann-Whitney U test ($P < 0.005$).

In our case, organizational metrics outperformed code churn metrics for both performance measures. Therefore, our findings confirm the potential benefit of organizational metrics for defect prediction. However, pre-release metrics (with 71% recall and 99% precision) outperformed all of the other metric sets including the organizational metrics. In this regard, the performance ranking of the metric sets is not consistent with the empirical findings of the original paper.

VI. Threats to Validity

In this section we discuss the threats to the validity of our replication that are specific to the replication study as well as the ones inherited from the original study.

Internal validity: Similar to the original study this replication is prone to issues related with the causal inferences. In terms of data quality, our data collection is a post-mortem activity, and, therefore it could not have been modified by the developers to affect the results. As for the operationalization of the constructs, we have strictly followed the original study and accordingly did not introduce any researcher bias that could have caused *expected* results. Though one of the authors of this paper was formerly affiliated with the case company, we have taken further precautions to mitigate the risk of researcher bias and we delegated the process of data collection and data analysis to the other authors of this paper.

Construct validity: In measuring the theoretical constructs defined in the original study, we followed a similar process defined therein by automating the data collection process through source code analyses, code revision systems and issue repositories. Since the organizational structure was not readily available to us, we had to regenerate it in a bottom-up fashion as explained in Section IV. In order to mitigate the risks, we have discussed and finalized the resulting structure with people from the case company who were able to cross check the outcome with the company yellow pages.

External validity: Similar to the original study, our results are based on the data collected from one software product of a specific company and may not reflect generalization to all software organizations. However, our replication itself is an effort towards enhancing the external validity of the original study. The different results of this replication are likely to be based on either the differences in the replication settings or the differences between the organizational contexts. In either

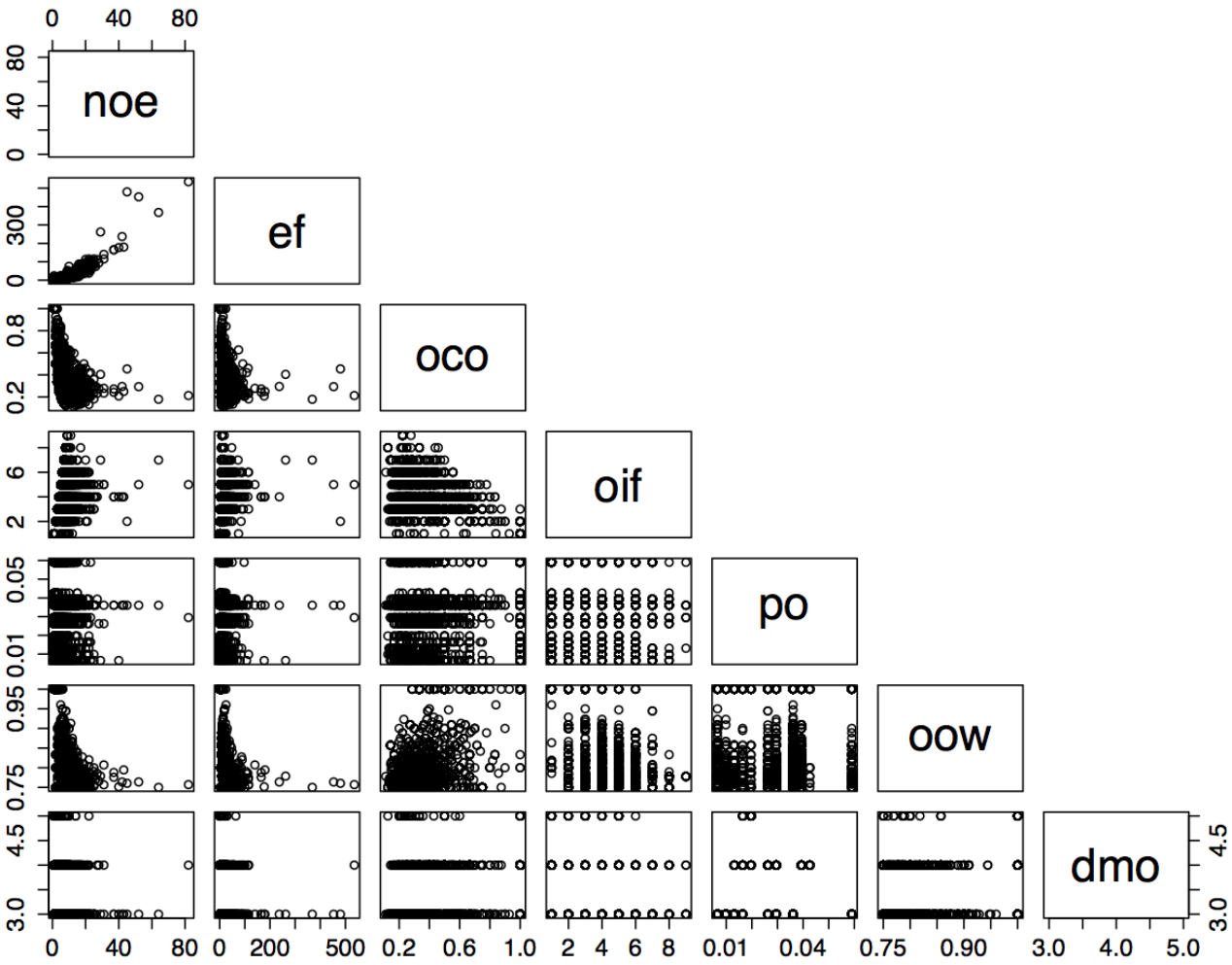


Fig. 1: Pairwise Scatter Plot of The Organizational Metrics.

case, more controlled replications are needed to be able to generalize the results or to scope them in specific settings.

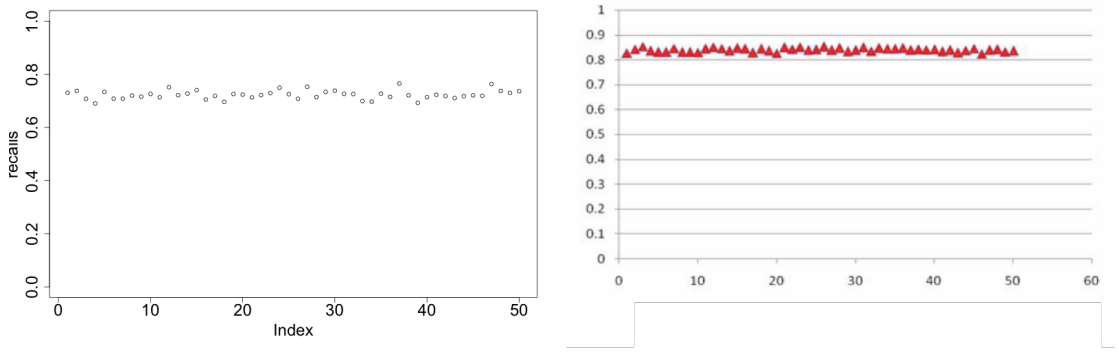
VII. Conclusions

In this paper, we replicated the work by Nagappan et al. on the effects of organizational metrics in software quality, pursuing the answers for the identification of organisational structure measures and their influence on and power to predict product quality [19]. Our replication was carried out in a different industrial environment with slight changes in the context.

Nagappan et al. proposed a new metric set reflecting organisational structure and found it to be superior over code-churn, complexity and pre-release metrics in predicting quality. In our replication, organizational metrics were not the top performing metric set; it was outperformed by a pre-release bug metric based prediction model. However, we verified that organisational metrics were better than code-churn metric set in predicting defects. In terms of the individual metrics in the proposed organisational metrics set, we verified the applicability for the four of them (NOE, EF, PO, OOW) and showed that the rest does not apply to our dataset.

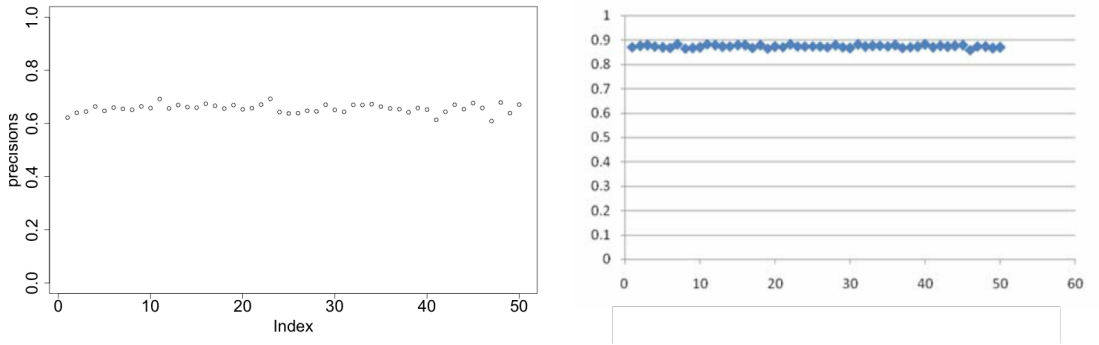
The changes in our empirical design, such as the difference in the prediction targets may have caused the differences in the results. Specifically, our functional method level predictions - as opposed to the binary file level predictions of the original study - is most likely the reason for observing pre-release bug metrics as the best indicators, as they represent a finer grained prediction level for the dependent variable, i.e., defects. Nevertheless, these changes in our study design allowed us to provide evidence for the validity and limitation of the original results in a different context.

Based on our findings we conclude that organizational metrics are worth further investigation, but we can not claim that they are the best indicators of defect proneness of a software product in all contexts, as we have demonstrated a counter-case. We encourage other researchers to replicate



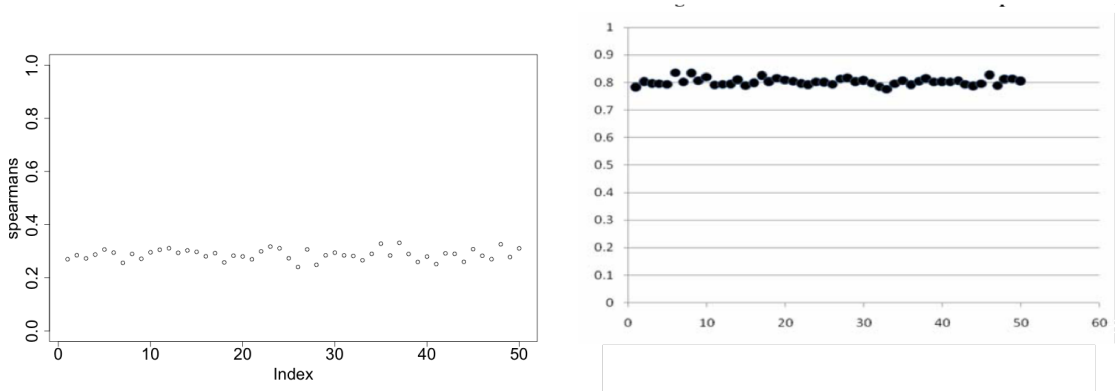
(a) Our Replication (b) Original Study [19]

Fig. 2: Comparison of Recall Values For 50 random splits for organizational metrics.



(a) Our Replication (b) Original Study [19]

Fig. 3: Comparison of Precision Values for 50 random splits for organizational metrics



(a) Our Replication (b) Original Study [19]

Fig. 4: Comparison of Spearman Correlation Values For 50 random splits for organizational metrics.

TABLE VI: Comparison of Prediction Performance for Different Metric Sets

Replication			**Original Study**		
Model	Precision	Recall	Model	Precision	Recall
Organizational Structure	66.2%	73.1%	Organizational Structure	86.2%	84.0
Code Churn	62.9%	61.6%	Code Churn	78.6%	79.9%
Code Complexity	61%	80.9%	Code Complexity	79.3%	66.0%
Pre-release bugs	70.8%	99.0%	Pre-release bugs	73.8%	62.9%
			Dependencies	74.4%	69.9%
			Code Coverage	83.8%	54.4%

either the original or our work in other datasets, especially in large-scale industrial projects, for external validation and generalization of the results.

ACKNOWLEDGMENTS

This research is supported partially by TEKES N4S programme in Finland and in Canada.

REFERENCES

[1] S. Alhassan, B. Caglayan, and A. B. Bener. Do more people make the code more defect prone?: Social network analysis in oss projects. In *SEKE*, pages 93–98, 2010.

[2] V. R. Basili, F. Shull, and F. Lanubile. Building knowledge through families of experiments. *Software Engineering, IEEE Transactions on*, 25(4):456–473, 1999.

[3] N. Bettenburg and A. E. Hassan. Studying the impact of social interactions on software quality. *Empirical Software Engineering*, 18(2):375–431, 2013.

[4] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. Does distributed development affect software quality?: an empirical case study of windows vista. *Communications of the ACM*, 52(8):85–93, 2009.

[5] C. Bird, N. Nagappan, B. Murphy, H. Gall, and P. Devanbu. Don't touch my code!: examining the effects of ownership on software quality. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 4–14. ACM, 2011.

[6] A. Brooks, J. Daly, J. Miller, M. Roper, and M. Wood. Replication of experimental results in software engineering. *International Software Engineering Research Network (ISERN) Technical Report ISERN-96-10, University of Strathclyde*, 1996.

[7] B. Caglayan, A. B. Bener, and A. Miranskyy. Emergence of developer teams in the collaboration network. In *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on*, pages 33–40. IEEE, 2013.

[8] J. Carver. Towards reporting guidelines for experimental replications: A proposal. In *Proceedings of the 1st International Workshop on Replication in Empirical Software Engineering Research*, (RESER) [Held during ICSE'10], Cape Town, South Africa, 2010.

[9] M. Cataldo and S. Nambiar. On the relationship between process maturity and geographic distribution: an empirical analysis of their impact on software quality. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 101–110. ACM, 2009.

[10] M. Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.

[11] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell. A systematic literature review on fault prediction performance in software engineering. *IEEE Trans. Softw. Eng.*, 38(6):1276–1304, Nov. 2012.

[12] J. D. Herbsleb and A. Mockus. An empirical study of speed and communication in globally distributed software development. *Software Engineering, IEEE Transactions on*, 29(6):481–494, 2003.

[13] M. Host, C. Wohlin, and T. Thelin. Experimental context classification: incentives and experience of subjects. In *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pages 470–478. IEEE, 2005.

[14] M. Jiménez, M. Piattini, and A. Vizcaíno. Challenges and improvements in distributed software development: A systematic review. *Advances in Software Engineering*, 2009:3, 2009.

[15] N. Juristo, A. M. Moreno, and S. Vegas. Reviewing 25 years of testing technique experiments. *Empirical Software Engineering*, 9(1-2):7–44, 2004.

[16] B. A. Kitchenham, T. Dyba, and M. Jorgensen. Evidence-based software engineering. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 273–281. IEEE, 2004.

[17] A. Meneely, L. Williams, W. Snipes, and J. Osborne. Predicting failures with developer networks and social network analysis. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 13–23. ACM, 2008.

[18] T. Menzies, B. Caglayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan. The promise repository of empirical software engineering data, June 2012.

[19] N. Nagappan, B. Murphy, and V. Basili. The influence of organizational structure on software quality: an empirical case study. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE'08, pages 521–530, New York, NY, USA, 2008. ACM.

[20] C. B. Seaman. Qualitative methods in empirical studies of software engineering. *Software Engineering, IEEE Transactions on*, 25(4):557–572, 1999.

[21] E. Shihab, C. Bird, and T. Zimmermann. The effect of branching strategies on software quality. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 301–310. ACM, 2012.

[22] F. Shull. Research 2.0? *Software, IEEE*, 29(6):4–8, Nov 2012.

[23] F. Shull, V. Basili, J. Carver, J. C. Maldonado, G. H. Travassos, M. Mendonça, and S. Fabbri. Replicating software engineering experiments: addressing the tacit knowledge problem. In *Empirical Software Engineering, 2002. Proceedings. 2002 International Symposium n*, pages 7–16. IEEE, 2002.

[24] D. I. Sjoberg, T. Dyba, and M. Jorgensen. The future of empirical methods in software engineering research. In *Future of Software Engineering, 2007. FOSE'07*, pages 358–378. IEEE, 2007.

[25] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer, 2012.

APPENDIX

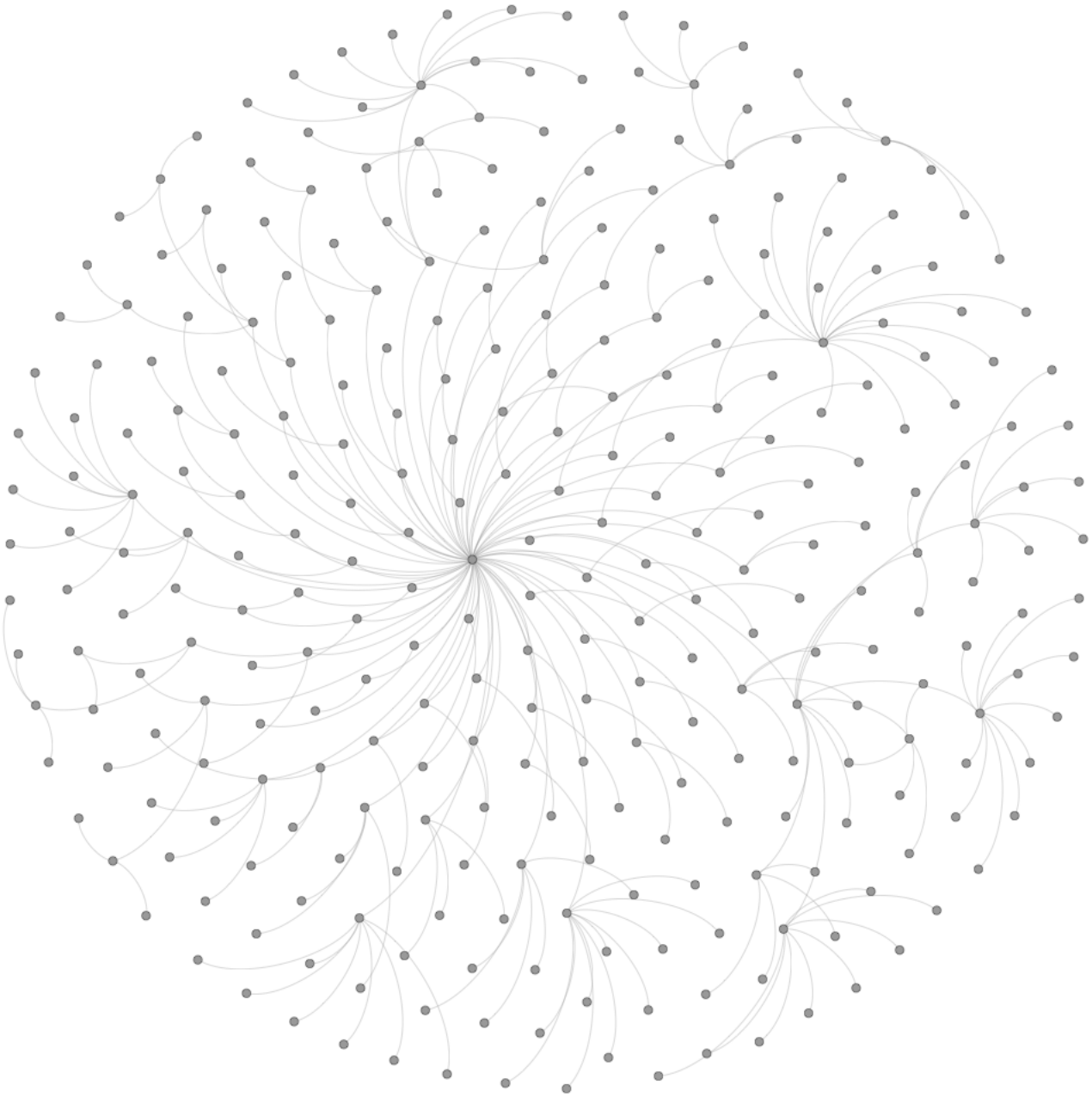


Fig. 5: A partial organizational hierarchy derived from the reporting structure among the contributors of the project investigated in this study. Each node corresponds to a person. The nodes at the periphery are the developers/ engineers and the inner nodes indicate more managerial responsibilities. We cannot report the actual organigram due to confidentiality issues. However, the number of people in the structure are 30+ and they are organised in 4+ levels, satisfying the minimum criteria set by the original study.