The Journal of Systems and Software 118 (2016) 288–296

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

# Effect of developer collaboration activity on software quality in two large scale projects

Bora Çaglayan*, Ayşe Başar Bener

*Mechanical and Industrial Engineering Department, Ryerson University, Toronto M5B 2K3, Canada*

ARTICLE INFO

*Article history:*
Received 23 April 2015
Revised 20 March 2016
Accepted 21 March 2016
Available online 31 March 2016

*Keywords:*
Collaboration networks
Developer collaboration
Software quality
Human factor in software engineering

ABSTRACT

Developers work together during software development and maintenance to resolve issues and implement features in large software projects. The structure of their development collaboration activity may have impact on the quality of the final product in terms of higher number of defects. In this paper, we aim to understand the effect of collaboration on the defect proneness software. We model the collaboration of developers as an undirected network. We extract the centrality of the developers from the collaboration network using different measures that quantifies the importance of the nodes. We analyze the defect inducing and fixing data of the developers in two large software projects. Our findings in this study can be summarized as follows: (a) Centrality and source code change activity of developers in the collaboration network may change their defect induction rates i.e. the defect proneness of their change sets, (b) Contrary to the common perception, more experienced people have relatively higher defect induction rates.

## 1. Introduction

Development and maintenance of large software is beyond the capacity of single person. Therefore most of the large software are built through the collaboration of many developers. We can observe the effort of individuals easily by mining source code repositories.

Coordination and management of the developers who may be working at distributed locations is a challenging activity. It may not be possible to observe collaborations directly due to unforeseen dependencies among the modules of the software in a given project. Lastly, changes in the workforce during development disrupt the structure of the collaboration. Dynamic developer collaboration structure can be extracted accurately only by mining the version control systems and analyzing the activity of the developers. The structure of the developer collaboration may depend on the architecture of the software, the organizational structure and the planned collaboration structure may change during the evolution of the software.

Collaboration structure of developers may have a complex relation with the quality of the software. In theory, cohesive teams with low inter-dependencies between each other are more likely to reduce their defect density Brooks (1995). However, on the source code level, collaboration network structure may be significantly different than the organizational structure Caglayan et al. (2013). Code level collaboration can be traced by mining the source code repositories. The experienced people among the developers tend to collaborate with more people over time. On the other hand, a nearly complete collaboration network may be a sign of collective code ownership advocated by the proponents of the agile methodologies (Beck, 2000).

Empirically, it has previously been observed that collaboration structure may be significantly different than the organizational structure. Cataldo et al. studied the effect of team structure on software quality previously (Cataldo and Herbsleb, 2010), (Caglayan et al., 2013). They defined teams based on the collaboration activity. In this aspect, one of their conclusions was the dramatic differences between the organizational team structure and the actual code-level collaboration structure.

In this paper, we have analyzed the relation of the collaboration network of the developers and the quality of the software modules. In our study, we assumed that the quality of a software module can be estimated by analyzing their defect proneness. We modeled software quality at change level by inspecting defect inducing and defect fixing changes. The research question that we address by this empirical study is as follows:

- **Research question:** What is the effect of development collaboration activity on the software quality?

To answer the research question, we extracted the collaboration activity and change categories for a large commercial enterprise

* Corresponding author. Tel.: +16479093489.
*E-mail addresses:* bora.caglayan@ryerson.ca (B. Çaglayan), ayse.bener@ryerson.ca (A.B. Bener).

http://dx.doi.org/10.1016/j.jss.2016.03.055
0164-1212/© 2016 Elsevier Inc. All rights reserved.

software and open source Eclipse Project. We examined the effect of collaboration on the software quality on change level. We grouped changes into defect inducing changes based on the methodology proposed by Śliwerski et al. (2005), defect fixing changes and normal changes. We modeled the collaboration activity as a collaboration network of developers based on co-work on similar software modules. Then, we checked the relation between the defect-inducing and defect-fixing activities and developer collaboration by using collaboration network metrics.

The contributions of this study are as follows:

1. We empirically studied the collaboration activity for two large-scale software.
2. We identified the effect of the centrality of developers on defect induction rates.

The structure of the rest of the paper is as follows: In the related work section we overview the relevant literature. In the methodology section we define our dataset, data extraction process and the empirical study setup. In the results section we present our empirical findings. Afterwards, we show the threats to the validity of the findings. Finally, in the conclusion section we present a summary of the important results and possible future work on this topic.

## 2. Related work

Programmer collaboration network and the effect of programmer collaboration on software quality has been investigated by several research groups. Herbsleb et al. investigated the relations between distributed software teams, increased software development costs and reduced productivity (Herbsleb and Mockus, 2003). He also examined if works in different sites can be interdependent and how may the interdependence diminish over time. The data from two multi-site companies have been used as input. Data was collected by surveys and from change management systems. Majority of the respondents point to difficulty of communication between sites. The two surveyed companies could not reduce interdependence of tasks between different sites in the projects. As a result they found that teams located at different sites have reduced overall efficiency by examining the software changes.

In a recent paper, Joblin et al. analyzed the developer networks of ten open source software Joblin et al. (2015). In this study, they have confirmed with 53 developers that the developer networks extracted from source code repositories automatically can model the collaboration.

Caglayan et al. 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 (Caglayan et al., 2013). They found that collaboration teams among the developers may form over time independent of the formal team structure of the organization.

Another notable paper in this area is a recent paper by Bettenburg et al. (Bettenburg, 2012). Bettenburg et al. 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. (Meneely et al., 2008) (Shin et al., 2011), Pinzger et al. Pinzger et al. (2008) and Alhassan et al. (Alhassan et al., 2010) tested the merits of the local collaboration metrics in the defect prediction problem. They found that the local collaboration metrics forms a strong alternative to the other well-established metric sets. Nagappan et al. proposed organizational metrics to predict the defect proneness of software (Nagappan et al., 2008). In their work, Nagappan et al. extracted metrics to understand the relations between organizational structure of a software company and the defect proneness of software modules.One example of the metrics they used is the percentage of organizational units which participated in the development of a software module.

Although the collaboration network idea has been used in multiple studies, to the best of our knowledge the effect of collaboration on software quality has not been studied in detail previously. Our work is different than the previous work that investigated the relation between developer collaboration and software quality in several ways: (1) We investigate the relation between developer collaboration and software quality on source code change level. Source code changes can be used to highlight the risky points in the software real-time. (2) We check the defect induction rates and its relation with developer collaboration for two large software, (3) We analyze the possible reasons of the effect of collaboration on software quality.

## 3. Methodology

### 3.1. Dataset

In our research, we used a large scale commercial software as well as a large scale open source software as a data source to check the effects of different licensing paradigms on our results.

*Enterprise software*

The first dataset that we have used is a commercial large-scale enterprise software product. The enterprise software product has a 20 year old code base. We examined a 500 kLOC part of the product that constitutes a set of architectural functionality of the project as the dataset. The programming languages of the project are C and C++. The software is developed by an international group of developers in five different countries.

*Eclipse project*

Eclipse project is a widely used multi-language software development platform written mainly in the Java programming language (Eclipse project web site, 2013). Its source code is forked from IBM VisualAge IDE. The project was made open-source in November 2001 and it was licensed initially under creative-commons license and later Eclipse Public License. Both of these licences are compatible with the FSF standards for open source licences. In our study we used the base IDE functionality component of Eclipse to construct the dataset (eclipse.platform.runtime). Within the span of our examination of the Eclipse project, most of the developers were IBM employees. According to (Capra et al., 2008) this categorizes Eclipse as a strictly governed commercial open source project.

### 3.2. Data extraction

We extracted the change history of the Eclipse and the Enterprise projects. Changes for the Eclipse Project. We used function change level granularity for the Enterprise software since it contained very large source code files ranging between 1 kLOC up to 15 kLOC. On the other hand, after a manual inspection we found that there is a lot of getter/setter functions in Eclipse without much content. We decided to use file change level granularity in Eclipse and function change level granularity in the Enterprise software for these reasons. The Enterprise Project had 6548 function changes during one major release while Eclipse project had 41,140 source code file changes between January 2001 and September 2013.