Information and Software Technology 64 (2015) 102–112

Contents lists available at ScienceDirect

Information and Software Technology

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

On the usefulness of ownership metrics in open-source software projects

Matthieu Foucault [a,*], Cédric Teyton [a], David Lo [b], Xavier Blanc [a], Jean-Rémy Falleri [a]

[a] University of Bordeaux, LaBRI, UMR 5800, F-33400 Talence, France
[b] School of Information Systems, Singapore Management University, Singapore

ARTICLE INFO

Article history:
Received 1 September 2014
Received in revised form 15 January 2015
Accepted 30 January 2015
Available online 9 February 2015

Keywords:
Software engineering
Empirical study
Process metrics

ABSTRACT

Context: Code ownership metrics were recently defined in order to distinguish major and minor contributors of a software module, and to assess whether the ownership of such a module is strong or shared between developers.
Objective: The relationship between these metrics and software quality was initially validated on proprietary software projects. Our objective in this paper is to evaluate such relationship in open-source software projects, and to compare these metrics to other code and process metrics.
Method: On a newly crafted dataset of seven open-source software projects, we perform, using inferential statistics, an analysis of code ownership metrics and their relationship with software quality.
Results: We confirm the existence of a relationship between code ownership and software quality, but the relative importance of ownership metrics in multiple linear regression models is low compared to metrics such as the number of lines of code, the number of modifications performed over the last release, or the number of developers of a module.
Conclusion: Although we do find a relationship between code ownership and software quality, the added value of ownership metrics compared to other metrics is still to be proven.

1. Introduction

Process metrics, which measure developer's activity, were shown to have a strong relationship with software quality and, to be more useful than code metrics when it comes to defect prediction [1]. Among process metrics, the ones introduced by Bird et al. that measure code ownership (CO) are of a particular interest [2]. These metrics, called *CO metrics* in this paper, quantify the level to which developers own modules of a software project, by measuring the ratio of contributions they make to such modules. CO metrics split developers of a module into two distinct groups: major and minor developers, who perform more and less than 5% of the contributions, respectively.

The usefulness of these metrics was validated on Microsoft software projects, showing that they have a strong relationship with the number of bugs of a module, and that adding code ownership metrics to a regression model (with the number of bugs as the dependent variable) improves its quality [2]. Bird et al. also observed that the more minor developers contribute to a software module, the more bugs it contains. A possible explanation comes from the fact that minor developers have less knowledge of the modules they contribute to, and therefore may introduce more bugs. Moreover, Bird et al. also observed that for a given software module two other metrics are related to its number of bugs: the number of major developers, and the ratio of contributions preformed by the main developer of a module to the total amount of contributions on such module. Contrary to minor developers, major developers have more insight on the modules they contribute to, and therefore may introduce less bugs.

Such a finding has two main consequences. First, development team should be reorganized with the objective to increase code ownership by limiting the number of minor developers, or if it is not possible, to have major developers reviewing the contributions of the minor ones. Second, CO metrics should be used when predicting the number of bugs of software modules, as adding them to a model significantly improves its quality.

As these results were observed solely on two Microsoft projects, we therefore replicated the Bird et al. study but with open-source software systems [3]. However, our replication, made on seven open-source Java software projects, did not yield the same observations. In particular, we did not observe any significant

* Corresponding author.
E-mail addresses: mfoucaul@labri.fr (M. Foucault), cteyton@labri.fr (C. Teyton), davidlo@smu.edu.sg (D. Lo), xblanc@labri.fr (X. Blanc), falleri@labri.fr (J.-R. Falleri).

http://dx.doi.org/10.1016/j.infsof.2015.01.013
0950-5849/

correlations between the CO metrics and the number of post-release bugs. So far, our replication was not complete as we only observed Java open-source software projects.

We therefore propose in this paper a deeper study that goes further and that aims to generally question the usefulness of the CO metrics for open-source software systems.

First of all, to overcome the limitation of our previous study, we propose in this study a new dataset of open-source software projects developed in several programming languages. Another essential point strengthening the validity of our study is the technique used to collect bug-related information: based on previous research, we concluded that automatic techniques developed to measure the number of bugs per module are not accurate nor precise enough [4–6], and therefore relied only on manually crafted data.

Further, we push our investigation toward the relative importance of the CO metrics for estimating the number of bugs. Our previous study only tries to observe a correlation between CO metrics and number of bugs, and does not investigate on the importance of these metrics in a model accounting for several variables. In this study we check their relative importance as compared to metrics that are frequently used to measure the quality of a software module, using an automatic technique called PMVD [7], which evaluates the importance of each metric in a multiple linear regression model, with the number of bugs as the dependent variable.

In comparison to our previous study, we therefore propose the new following contributions:

- A completely new dataset that contains open-source projects developed in different programming languages, and manually crafted bug-related data.
- New results of correlation between CO metrics and post release bugs.
- An investigation on the relative importance of CO metrics.

This paper is structured as follows: Section 2 presents the foundations of code ownership and the metrics related to it. Section 3 presents the detailed methodology of our study, including the construction of the dataset. Section 4 presents the main results of our study which shows that the usefulness of CO metrics is debatable in case of open-source software systems. Section 5 presents the threats to the validity of our study. Finally, Section 6 provides an overview of the related work and Section 7 concludes this paper.

2. Background and theory

This section starts by presenting the code ownership (CO) metrics that have been defined to measure to which extent developers own software modules.

2.1. Ownership metrics

Before explaining how CO metrics are measured, we need to define the model we use to represent a software development project and define the pertinent concepts, such as software module and developer contribution.

We assume that a software project is composed of a finite set of software modules that are developed by a finite set of developers who submit their code modifications by sending commits to a shared code repository.

Each module is defined by a finite set of source code files. When a developer modifies one of the files of a software module by committing her work, she is contributing to that module. The weight of the contribution made by a developer to a given module can be measured with different metrics. Bird et al. [2] chose to measure it by counting the number of files touched by the developer. For example, if Alice contributes to a module by modifying three files in a first commit and five files afterwards, she is contributing with a weight of eight. Another possibility is to measure the weight of a developer contribution by counting the number of line changes performed by the developer, also called code churn [8].

In our formal definitions, we use D as the set of developers that contribute to the project. For a given module, we define w_d as the weight of a developer d.

CO metrics mainly measure the ratio of contributions made by one developer compared to the rest of the team. More formally, for a given module, the ownership of a developer d is:

$$own_d = \frac{w_d}{\sum_{d' \in D}(w_{d'})}$$

Bird et al. [2] proposed three ownership-based metrics that are computed for each software module:

Most valued owner[1] This score is the highest value of the ratio of contributions performed by all developers. More formally, for a given software module, its *Most valued owner* value (MVO) is

$$\max(\{own_d | d \in D\})$$

Minor This score counts how many developers have a ratio of contributions that is lower than 5%. Such developers are considered to be minor contributors of the software module. More formally, for a given software module, its *Minor* value is

$$|\{0 < own_d \leqslant 5\% | d \in D\}|$$

Major This score counts how many developers have a ratio of contributions that is bigger than 5%. Such developers are considered to be major contributors of the software module. More formally, for a given software module, its *Major* value is

$$|\{own_d > 5\% | d \in D\}|$$

Bird et al. showed that varying the 5% threshold used by the metrics *Minor* and *Major* to other values from 2% to 10% did not impact the results they obtained regarding the relationship between code ownership and software quality.

2.2. Code ownership and software quality

When the amount of developers of a software system rises, work must be divided between contributors. Whether a shared or strong ownership is preferable is a matter of debate where two theories come face to face. On the one hand the XP movement [9] and Raymond [10] advocate shared ownership, and the latter introduced "Linus' Law", which states that "given enough eyeballs, all bugs are shallow", i.e., increasing the number of contributors accelerates the detection and correction of bugs. On the other hand Bird et al. [2]. advocate for a strong ownership, and aim to confirm the "too many cooks spoil the broth" theory stating that when the number of developers increases, coordination in the development efforts becomes too complex to ensure. Further, both theories are backed by empirical findings: Rahman and Devanbu [11], considered ownership at the level of individual lines of code, and found that code implicated in bugs was strongly associated to a single developer's contribution.

In this paper we focus on the second theory, and deeply investigate on the Bird et al. CO metrics. As these metrics were only validated on Microsoft software system, we here check their

[1] This metric originally called *ownership* has been renamed here for sake of clarity.