Contents lists available at ScienceDirect

Information and Software Technology

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

An empirical study on software defect prediction with a simplified metric set

Peng He [a,b], Bing Li [c,d], Xiao Liu [a,b], Jun Chen [b,e], Yutao Ma [b,d,*]

[a] State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China
[b] School of Computer, Wuhan University, Wuhan 430072, China
[c] International School of Software, Wuhan University, Wuhan 430079, China
[d] Research Center for Complex Network, Wuhan University, Wuhan 430072, China
[e] National Engineering Research Center for Multimedia Software, Wuhan University, Wuhan 430072, China

ARTICLE INFO

Article history:
Received 28 February 2014
Received in revised form 18 November 2014
Accepted 20 November 2014
Available online 28 November 2014

Keywords:
Defect prediction
Software metrics
Metric set simplification
Software quality

ABSTRACT

Context: Software defect prediction plays a crucial role in estimating the most defect-prone components of software, and a large number of studies have pursued improving prediction accuracy within a project or across projects. However, the rules for making an appropriate decision between within- and cross-project defect prediction when available historical data are insufficient remain unclear.
Objective: The objective of this work is to validate the feasibility of the predictor built with a simplified metric set for software defect prediction in different scenarios, and to investigate practical guidelines for the choice of training data, classifier and metric subset of a given project.
Method: First, based on six typical classifiers, three types of predictors using the size of software metric set were constructed in three scenarios. Then, we validated the acceptable performance of the predictor based on Top-*k* metrics in terms of statistical methods. Finally, we attempted to minimize the Top-*k* metric subset by removing redundant metrics, and we tested the stability of such a minimum metric subset with one-way ANOVA tests.
Results: The study has been conducted on 34 releases of 10 open-source projects available at the PROMISE repository. The findings indicate that the predictors built with either Top-*k* metrics or the minimum metric subset can provide an acceptable result compared with benchmark predictors. The guideline for choosing a suitable simplified metric set in different scenarios is presented in Table 12.
Conclusion: The experimental results indicate that (1) the choice of training data for defect prediction should depend on the specific requirement of accuracy; (2) the predictor built with a simplified metric set works well and is very useful in case limited resources are supplied; (3) simple classifiers (e.g., Naïve Bayes) also tend to perform well when using a simplified metric set for defect prediction; and (4) in several cases, the minimum metric subset can be identified to facilitate the procedure of general defect prediction with acceptable loss of prediction precision in practice.

1. Introduction

In software engineering, defect prediction can precisely estimate the most defect-prone software components, and help software engineers allocate limited resources to those bits of the systems that are most likely to contain defects in testing and maintenance phases. Understanding and building defect predictors (also known as defect prediction models) for a software project is useful for a variety of software development or maintenance activities, such as assessing software quality and monitoring quality assurance (QA).

The importance of defect prediction has motivated numerous researchers to define different types of models or predictors that characterize various aspects of software quality. Most studies usually formulate such a problem as a supervised learning problem, and the outcomes of those defect prediction models depend on historical data. That is, they trained predictors from the data of historical releases in the same project and predicted defects in the upcoming releases, or reported the results of cross-validation on the same data set [16], which is referred to as Within-Project Defect Prediction (WPDP). Zimmermann et al. [3] stated that defect

* Corresponding author at: School of Computer, Wuhan University, Wuhan 430072, China. Tel.: +86 27 68776081.

E-mail addresses: penghe@whu.edu.cn (P. He), bingli@whu.edu.cn (B. Li), lxiao@whu.edu.cn (X. Liu), chenj@whu.edu.cn (J. Chen), ytma@whu.edu.cn (Y. Ma).

http://dx.doi.org/10.1016/j.infsof.2014.11.006
0950-5849/© 2014 Elsevier B.V. All rights reserved.

prediction performs well within projects as long as there is a sufficient amount of data available to train any models. However, it is not practical for new projects to collect such sufficient historical data. Thus, achieving high accuracy defect prediction based on within-project data is impossible in some cases.

Conversely, there are many public on-line defect data sets available, such as PROMISE,[1] Apache[2] and Eclipse.[3] Some researchers have been inspired to overcome this challenge by applying the predictors built for one project to a different one [3,17,65]. Utilizing data across projects to build defect prediction models is commonly referred to as Cross-Project Defect Prediction (CPDP). CPDP refers to predicting defects in a project using prediction models trained from the historical data of other projects. The selection of training data depends on the distributional characteristics of data sets. Some empirical studies evaluated the potential usefulness of cross-project predictors with a number of software metrics (e.g., static code metrics, process metrics, and network metrics) [15,16] and how these metrics could be used in a complementary manner [8]. Unfortunately, despite these attempts to demonstrate the feasibility of CPDP, this method has been widely challenged because of its low performance in practice [15]. Moreover, it is still unclear how defect prediction models between WPDP and CPDP are rationally chosen when limited or insufficient historical data are provided.

In defect prediction literature, a considerable number of software metrics, such as static code metrics, code change history, process metrics and network metrics [1,4–7,10], have been used to construct different predictors for defect prediction [35]. Almost all existing prediction models are built on the complex combinations of software metrics, with which a prediction model usually can achieve a satisfactory accuracy. Although some feature selection techniques (e.g., principal component analysis (PCA)) successfully reduce data dimensions [2,44–46,50], they still lead to a time-consuming prediction process. Can we find a compromise solution that makes a tradeoff between cost and accuracy? In other words, can we find a universal predictor built with few metrics (e.g., Lines Of Code (LOC)) that achieves an acceptable result compared with those complex prediction models?

In addition to the selection of a wide variety of software metrics, there are many classifiers (learning algorithms) that have been studied, such as Naïve Bayes, J48, Support Vector Machine (SVM), Logistic Regression, and Random Tree [24,27,29], and defect prediction using these typical classifiers has achieved many useful conclusions. Currently, some improved classifiers [26,54,65] and hybrid classifiers [30–32] have also been proposed to effectively improve classification results. Menzies et al. [33] advocated that different classifiers have indiscriminate usage and must be chosen and customized for the goal at hand.

Fig. 1 presents a summary of the state-of-the-art defect prediction. Complex predictors improve prediction precision with loss of generality and increase the cost of data acquisition and processing. On the contrary, simple predictors are more universal, and they reduce the total effort-and-cost by sacrificing a little precision. To construct an appropriate and practical prediction model, we should take into overall consideration the precision, generality and cost according to specific requirements. Unlike the existing studies on complex predictors, in our study, we focus mainly on building simple prediction models with a simplified metric set according to two assumptions (see the contents with a gray background in Fig. 1), and seek empirical evidence that they can achieve acceptable results compared with the benchmark models. Our contributions to the current state of research are summarized as follows:

- We proposed an easy-to-use approach to simplifying the set of software metrics based on *filters* methods for feature selection, which could help software engineers build suitable prediction models with the most representative code features according to their specific requirements.
- We also validated the optimistic performance of the prediction model built with a simplified subset of metrics in different scenarios, and found that it was competent enough when using different classifiers and training data sets from an overall perspective.
- We further demonstrated that the prediction model constructed with the minimum subset of metrics can achieve a respectable overall result. Interestingly, such a minimum metric subset is stable and independent of the classifiers under discussion.

With these contributions, we complement previous work on defect prediction. In particular, we provide a more comprehensive suggestion on the selection of appropriate predictive modeling approaches, training data, and simplified metric sets for constructing a defect predictor according to different specific requirements.

The rest of this paper is organized as follows. Section 2 is a review of related literature. Sections 3 and 4 describe the approach of our empirical study and the detailed experimental setups, respectively. Sections 5 and 6 analyze and discuss the primary results, and some threats to validity that could affect our study are presented in Section 7. Finally, Section 8 concludes the paper and presents the agenda for future work.

2. Related work

Defect prediction is an important topic in software engineering, which allows software engineers to pay more attention to defect-prone code with software metrics, thereby improving software quality and making better use of limited resources.

2.1. Within-project defect prediction

Catal [28] investigated 90 software defect prediction papers published between 1990 and 2009. He categorized these papers and reviewed each paper from the perspectives of metrics, learning algorithms, data sets, performance evaluation metrics, and experimental results in an easy and effective manner. According to this survey, the author stated that most of the studies using method-level metrics and prediction models were mostly based on machine learning techniques, and Naïve Bayes was validated as a robust machine learning algorithm for supervised software defect prediction problems.

Hall et al. [22] investigated how the context of models, the independent variables used, and the modeling techniques applied affected the performance of defect prediction models according to 208 defect prediction studies. Their results showed that simple modeling techniques, such as Naïve Bayes or Logistic Regression, tended to perform well. In addition, the combinations of independent variables were used by those prediction models that performed well, and the results were particularly good when feature selection had been applied to these combinations. The authors argued that there were a lot of defect prediction studies in which confidence was possible, but more studies that used a reliable methodology and that reported their detailed context, methodology, and performance in the round were needed.

The vast majority of these studies were investigated in the above two systematic literature reviews that were conducted in the context of WPDP. However, they ignored the fact that some projects, especially new projects, usually have limited or insufficient

[1] http://promisedata.org.
[2] http://www.apache.org/.
[3] http://eclipse.org.