Identification of Co-Changed Classes in Software Applications Using Software Quality Attributes

Anushree Agrawal, Indira Gandhi Delhi Technical University For Women, India

R. K. Singh, Indira Gandhi Delhi Technical University For Women, India

https://orcid.org/0000-0001-8729-2293

ABSTRACT

When changes are made to software applications often, defects can occur in software applications, and eventually leads to expensive operational faults. Comprehensive testing is not feasible with the limited time and resources available. There is a need for test case selection and prioritization so that testing can be completed with maximum confidence in a minimum time. Advance knowledge of co-changed classes in software applications can be very useful during the software maintenance phase. In this article, the authors have proposed a co-change prediction model based upon the combination of structural code measures and dynamic revision history from change repository. Univariate analysis is applied to identify the useful measures in co-change identification. The proposed model is validated using eight open source software applications. The results are promising and indicate that they can be very beneficial in maintenance of software applications.

KEYWORDS

Co-change Prediction, Open Source Software, Regression Analysis, Software Metrics

INTRODUCTION

Complexity of code interaction increases significantly with evolution in software applications. It becomes extremely challenging for the developers to control the impact of changes implemented in the system. Change impact analysis is a prospective solution to this problem, which identifies the entities affected by a particular change. The advance knowledge of these co-changed entities can be very useful in the software maintenance phase. This information will assist the developers in identifying the effects of changes made to a class and also help in testing phase by helping in test case selection and prioritization.

Identification of co-changed classes is very beneficial when only few classes in a software system change during the maintenance phase. It can identify dependencies and thus eliminate the need for testing the complete software system, which is quite tedious. It will guide the developers by providing the advance knowledge of dependent classes and also help the testers in choosing the appropriate set of changed entities thus, accelerating the testing phase.

In literature, very few studies focus on identifying the co-changed classes and use the information derived from previous logs for identifying the co-changed classes. The most common approach uses
static code attributes for identification of such classes (Ying, Murphy, Ng & Chu-Carroll, 2004; Zimmermann, Weibgerber, Diehl & Zeller, 2005). A limitation of this approach is that it is not able to identify any dependencies among software entities and unable to identify the potential set of classes which are likely to be changed due to modifications in a class (also called as the ripple effect). It is not able to identify any dependencies that may occur due to addition of new classes.

To overcome this limitation, we have proposed a predictive modeling-based solution that can identify the after-effects of newly added classes using evolutionary coupling. We have used the logs from software version control system (svn repository) to obtain the co-change information of classes and set of object-oriented software metric to establish the relationship among these co-changed classes. The combination of software metrics and revision history has been used in predicting the future changeability pattern of software entities.

The objective of this research is to investigate the evolutionary coupling among classes for prediction of future changeability pattern. It is believed that classes that have been changed together frequently in past are likely to be changed together in future as well (Zimmermann, Weibgerber, Diehl & Zeller, 2005; Parashar & Chhabra, 2014, Rolfsnes et al., 2017). We obtain this co-change information from the revision history and use it to predict evolutionary coupling among classes. We have used eight open source software applications for performing our experiments. All the software applications are written in Java and maintained under SVN repository. We have collected the metric information from the source code and change proneness from the logs obtained from SVN repository. In this work, we have addressed the following research questions:

RQ1. Is it possible to identify the classes that are modified as a ripple effect of change prone class?
RQ2. What are the possible attributes that can be useful for ripple effect identification of changed class?
RQ3. How well is the proposed system applicable to commercial systems?

The rest of the paper is organized as follows. Section 2 presents the related work. The research methodology is discussed in section 3 followed by empirical data collection process in section 4. The results and threats to validity are discussed in section 5. Section 6 concludes the paper and presents the future scope of work.

RELATED WORK

Changeability is an important attribute of maintainability and indicates the ease of modifying a software system. Change-coupling prediction plays an important role in changeability measurement. In literature two approaches for identifying change prone entities in software applications have been proposed (Zimmermann, Weibgerber, Diehl & Zeller, 2005; Lu, Zhou, Xu, Leung & Chen, 2012; Brucker & Julliand, 2014). One suggests the use of structural information of object-oriented systems and the other suggests mining past co-changes to predict future changeability pattern.

A lot of work has been done in literature for change proneness prediction using structural information (Lu, Zhou, Xu, Leung & Chen, 2012; Brucker & Julliand, 2014). One of the limitations of this approach is that it is not able to identify the co-changed classes in software applications (Brucker & Julliand, 2014). Lu et al. (2012) conducted a study on 102 java systems using 62 OO metrics to establish relationship between Object oriented metrics and change proneness in OO software applications. They found that size metrics exhibit moderate predictive ability; coupling and cohesion metrics have low predictive ability while inheritance metrics have poor predictive ability. Li et al. (2013) conducted a extensive study on code based Change Impact Analysis (CIA) techniques and suggest the need to evaluate existing CIA techniques and proposing new techniques, however several research questions still remain unanswered. E.g. what are the benchmarks for CIA technique.
ERP Implementation in State Government
www.igi-global.com/article/erp-implementation-state-government/44549?camid=4v1a