Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 62 (2015) 529 – 538

Procedia Computer Science

The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)

Predicting Unit Testing Effort Levels of Classes: An Exploratory Study based on Multinomial Logistic Regression Modeling

Mourad Badri[a], Fadel Toure[a,b] , Luc Lamontagne[b,*]

[a] *Department of Mathematics and Computer Science, University of Quebec, Trois-Rivières, Quebec, Canada*
[b] *Department of Computer Science and Software Engineering, Laval University, Quebec, Canada*

Abstract
The study aims at investigating empirically the ability of a *Quality Assurance Indicator* (Qi), a metric that we proposed in a previous work, to predict different levels of unit testing effort of classes in object-oriented systems. To capture the unit testing effort of classes, we used four metrics to quantify various perspectives related to the code of corresponding unit test cases. Classes were classified, according to the involved unit testing effort, in five categories (levels). We collected data from two open source Java software systems (ANT and JFREECHART) for which JUnit test cases exist. In order to explore the ability of the Qi metric to predict different levels of the unit testing effort of classes, we decided to explore the possibility of using the Multinomial Logistic Regression (MLR) method. The performance of the Qi metric has been compared to the performance of three well-known source code metrics related respectively to size, complexity and coupling. Results suggest that the MLR model based on the Qi metric is able to accurately predict different levels of the unit testing effort of classes.

Peer-review under responsibility of organizing committee of The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)

Keywords: Object-Oriented Systems; Software Testability; Unit Testing Effort; JUnit Code; Metrics; Prediction; Multinomial Logistic Regression; Empirical Analysis.

1. Introduction

Software testing plays a crucial role in software quality assurance. It is, however, a time and resources consuming process. The overall effort spent on testing depends, in fact, on many different factors (Yeh and Lin, 1998; Baudry et *al*., 2003-a; Baudry et *al*., 2003-b; Bruntink and Deursen, 2006) including human factors, process issues, testing techniques, tools used, characteristics of the software development artifacts, and so forth. We focus, in this paper, on unit testability of classes (unit test case construction), and particularly on the effort involved to write the code of unit test cases. A large number of object-oriented (OO) metrics were proposed in literature (Henderson-Seller, 1996). Some of these metrics, related to different OO internal class attributes (such as size, complexity, coupling, cohesion and inheritance), were already used in recent years to predict unit testability of classes in OO software systems (Bruntink and Deursen, 2004; Gupta et *al*., 2005; Bruntink and Deursen, 2004; Singh et *al*., 2008; Singh and Saha, 2010; Badri et *al*., 2010; Badri et *al*., 2011; Badri and Touré, 2011; Badri and Touré, 2012-a; Badri and Touré, 2012-b; Zhou et *al*., 2012). Unfortunately, little empirical evidence exists on the ability of these metrics to predict different levels of the unit testing effort of classes. As far as we know, this issue has not been

* Mourad Badri. Tel: +1 819 376 5011 ext. 3834; fax: +1 819 376 5185
E-mail address: Mourad.Badri@uqtr.ca

Peer-review under responsibility of organizing committee of The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)
doi:10.1016/j.procs.2015.08.528

empirically investigated. Classifying the software classes according to the required testing effort could, in fact, help managers to better plan and monitor testing activities and resources, and determine the critical classes requiring a relatively high testing effort on which developers and testers have to focus to ensure software quality.

We proposed in a previous work (Badri et *al*., 2009) a synthetic metric called *Quality Assurance Indicator* (Qi). Basically, the Qi of a class is based on intrinsic characteristics of the class as well as on the Qi of its collaborating classes. The metric has, however, no ambition to capture the overall quality (or testability) of OO software systems. Moreover, the objective is not to evaluate a design by giving absolute values, but more relative values that may be used for identifying, for example: (1) classes that will require a high testing effort, and (2) classes on which more testing effort is required to ensure software quality. We focus, in this paper, on the first objective. We explored in (Badri et *al*., 2011) the relationship between the Qi metric and unit testability of classes using correlations. Unit testability of classes was measured using two metrics (Bruntink and Deursen, 2004; Bruntink and Deursen, 2006): the *number of lines* of test code and the *number of assert statements* in the test code. Furthermore, we investigated in (Badri and Touré, 2011) the ability of the Qi metric to predict the unit testing effort using the Univariate Logistic Regression method. In this paper, we extend the study presented in (Badri and Touré, 2012-a) by investigating the ability of the Qi metric to predict various levels (categories) of the unit testing effort of classes. We collected data from two open source Java software systems (ANT and JFREECHART) for which JUnit test cases exist. To capture the unit testing effort of classes, we used this time four metrics to quantify various perspectives related to the code of corresponding JUnit test cases. Classes were classified, according to the involved unit testing effort, in five categories. In order to investigate the ability of the Qi metric to predict the different levels of the unit testing effort of classes, we used the Multinomial Logistic Regression (MLR) method. The MLR method is a generalization of the Logistic Regression (LR) method. MLR modeling is, in fact, used for data in which the dependent variable is unordered or polytomous (i.e. its values are more than two categories, which is our case), and independent variables are continuous or categorical predictors. The MLR method is more appropriate for our study than the simple LR analysis. The performance of the Qi metric has been compared to the performance of three source code metrics related respectively to three important internal class attributes: size, complexity and coupling. Results show that the MLR model based on the Qi metric is able to accurately predict different levels of the unit testing effort of classes.

The rest of this paper is organized as follows: Section 2 gives a summary on related work. The Qi metric is briefly presented in Section 3. Section 4 presents the unit test case metrics that we used in our study. Section 5 presents the empirical study that we conducted. Finally, Section 6 summarizes the contributions of this work and outlines some future work directions.

2. Related Work

Software testability (IEEE, 1990; ISO/IEC9126, 1991; Fenton and Pfleeger, 1997) is an important software quality attribute. Software testability is, in fact, a complex notion that is affected by many different factors. It has been addressed in literature from different point of views. We focus, in this paper, on unit testability of classes (unit test case construction), and particularly on the effort involved to write the code of unit test cases. Various OO metrics have been used in recent years to predict unit testability of classes in OO software systems. Bruntink and Van Deursen (Bruntink and Deursen, 2004; Bruntink and Deursen, 2006) investigated factors of testability of OO software systems and explored the relationships between OO design metrics and some characteristics of JUnit test cases. Testability was measured by the number of lines of test code and the number of assert statements in the test code. Singh et *al*. (Singh et *al*., 2008) used OO metrics and neural networks to predict the testing effort, measured in terms of lines of code added or changed during the lifecycle of a defect. Singh and Saha (Singh and Saha, 2010) focused on the prediction of the testability of Eclipse at the package level. Testability was measured using several metrics including the number of lines of test code, the number of assert statements in the test code, the number of test methods and the number of test classes. Badri et *al*. (Badri et *al*., 2010; Badri et *al*., 2011) investigated the ability of lack of cohesion metrics to predict unit testability of classes using LR methods. Badri and Touré (Badri and Touré, 2012-b) explored the ability of OO metrics to predict the unit testing effort of classes using LR analysis. In these