Source and Test Code Size Prediction
A Comparison between Use Case Metrics and Objective Class Points

Mourad Badri, Linda Badri and William Flageol
Software Engineering Research Laboratory, Department of Mathematics and Computer Science, University of Quebec, Trois-Rivières, Quebec, Canada

Keywords: Use Cases, Use Case Metrics, Class Diagrams, Objective Class Points, Source Code Size, Test Code Size, Prediction Models, Linear Regression.

Abstract: Source code size, in terms of SLOC (Source Lines of Code), is an important parameter of many parametric software development effort estimation methods. Moreover, test code size, in terms of TLOC (Test Lines of Code), has been used in many studies to indicate the effort involved in testing. This paper aims at comparing empirically the Use Case Metrics (UCM) method, a use case model based method that we proposed in previous work, and the Objective Class Points (OCP) method in terms of early prediction of SLOC and TLOC for object-oriented software. We used both simple and multiple linear regression methods to build the prediction models. An empirical comparison, using data collected from four open source Java projects, is reported in the paper. Overall, results provide evidence that the multiple linear regression model, based on the combination of the use case metrics, is more accurate in terms of early prediction of SLOC and TLOC than: (1) the simple linear regression models based on each use case metric, and (2) the simple linear regression model based on the OCP method.

1 INTRODUCTION

Software development is a time and resource consuming process. Furthermore, software testing, which plays a crucial role in software quality assurance, takes an important part of the software development effort. It is, therefore, important to estimate as soon as possible, ideally in the early stages of the software development lifecycle, the effort required to develop and test software. In this way, activities can be planned and resources can be optimally allocated in order to ensure a successful software development (and testing) management.

Many software development effort estimation methods have been proposed in the literature over the years (Albrecht, 1979; Albrecht and Gaffney, 1983; Antoniol et al., 1999; Antoniol et al., 2003; Bianco and Lavazza, 2006a; Bianco and Lavazza, 2006b; Boehm, 1981; Boehm et al., 2000a; Boehm et al., 2000b; Bourque and Côté, 1991; Carbone and Santucci, 2002; Carroll, 2005; Chen et al., 2004; Hastings and Sajeev, 2001; Henderson-Sellers, 1997; Jahan and Sheibani, 2005; Janaki and Raju, 2000; Jorgensen and Boehm, 2009; Jorgensen and Shepperd, 2007; Kemerer, 1987; Lagerstrom et al., 2012; Kim et al., 2006; Leung and Fan, 2002; Matson et al., 1994; McDonnel, 2003; Mishra et al., 2010; Ochodek et al., 2011; Pfleeger et al., 2005; Trendowicz et al., 2008; Verner and Tate, 2012; Zhao and Tan, 2003; Zhou et al., 2014). Source code size, in terms of SLOC (Source Lines of Code), is an important parameter of many parametric software development effort estimation methods. The testing effort is unfortunately often estimated as a part of the overall software development effort. However, test code size, in terms of TLOC (Test Lines of Code), has been used in many studies to indicate the effort involved in unit testing (Badri and Toure, 2012; Bruntik and Van Deursen, 2004; Bruntik and Van Deursen, 2006; Singh et al., 2008; Singh and Saha, 2010; Zhou et al., 2012).

In this paper, we focus on the early prediction of both SLOC and TLOC for object-oriented (OO) software development, which is used extensively in the industrial projects. The study aims basically at comparing empirically the Use Case Metrics (UCM) method, which is a use case model based prediction method that we proposed in a previous work (Badri et al., 2013), and the Objective Class

Points (OCP) method (Kim et al., 2005) in terms of early prediction of SLOC and TLOC. The UCM method has already been compared in (Badri et al., 2015) to the Use Case Points (UCP) method, which is also a use case model based estimation method. Results show that the UCM method is more accurate, in terms of predicting test suites size, than the UCP method. In this work, we wanted to compare the UCM method to a class diagram based method. We used both simple and multiple linear regression methods to build the prediction models. An empirical comparison, using data collected from four open source Java projects, is reported in the paper. Overall, results provide evidence that the multiple linear regression model, based on the combination of the use case metrics, is more accurate in terms of early prediction of SLOC and TLOC than: (1) the simple linear regression models based on each use case metric, and (2) the simple linear regression model based on the OCP metric.

The rest of this paper is organized as follows: Section 2 presents summarily the Objective Class Points method. The use case metrics are introduced in Section 3. Section 4 presents the different steps of the empirical study we conducted to evaluate comparatively the Use Case Metrics and the Objective Class Points approaches in terms of SLOC and TLOC prediction accuracy. Finally, Section 5 concludes the paper and outlines some future work directions.

2 OBJECTIVE CLASS POINTS METHOD

The Objective Class Points (OCP) method counts the number of class points in a UML class diagram (Kim et al., 2005). The OCP method, unlike other class points based approaches such as (Costagliola et al., 2005; Abrahao et al., 2006), does not involve any subjective factors (Zhou et al., 2014). For a class diagram, the OCP value is computed as follows (Kim et al., 2005; Zhou et al., 2014):

$$OCP = NC + NGen + NDep + NORR + NM + NA + NAssoc + NAgg + NComp$$

where: NC: the total number of classes, NGen: the total number of generalization relationships, NDep: the total number of dependency relationships, NORR: the total number of realization relationships, NM: the total number of methods, NA: the total number of attributes, NAssoc: the total number of association relationships, NAgg: the total number of aggregation relationships, and NComp: the total number of composition relationships.

3 USE CASE METRICS

A use case model defines the functional scope of the system to be developed (Jacobson et al., 1993; Larman, 2004). A use case is a collection of related success and failure scenarios that describe actors using a system to support a goal. Use cases describe how external actors interact with the software system. These interactions generate events to the software system, known as input system events, which are usually associated with system operations. A scenario, also called a use case instance, is a specific sequence of actions and interactions between actors and the system. It is one particular story of using the system, or one path through the use case. We present, in what follows, the use case metrics that we used in our study, metrics that we introduced in previous work (Badri et al., 2013; Badri et al., 2015), to quantify different characteristics related to size and complexity of use cases:

Number of Involved Classes (NIC): This metric defines the total number of analysis classes participating in a use case, which have been identified during the analysis phase.

Number of External Operations (NEO): This metric defines the total number of system operations associated with the system events related to a use case. These operations are easily identifiable from UML system sequence diagrams.

Number of Scenarios (NS): This metric gives the total number of the different scenarios of a use case. One use case can, in fact, have one or more scenarios. This metric is related to the cyclomatic complexity of the use case.

Number of Transactions (NT): This metric gives the total number of transactions of a use case. A transaction is an event (a set of activities in a use case scenario) that occurs between an actor and the target system, the event being performed entirely or not at all. The complexity of a use case is also determined by the number of transactions.

4 EMPIRICAL STUDY

4.1 Goals, Selected Projects and Data Collection

The study aims at comparing empirically the Use Case Metrics and the Objective Class Points approaches in terms of predicting both source code size (SLOC) and test code size (TLOC). For a use case (noted UC_i), to indicate: (1) the size of corresponding source code, we used the SLOC metric, and (2) the size of corresponding test code, we used the TLOC metric. The SLOC and TLOC metrics are defined as follows:

Source Lines Of Code (SLOC): This metric defines the cumulative number of lines of source code related to a use case (all of its scenarios). It is used to indicate the total size, in terms of lines of source code, of the parts of software corresponding to a use case.

Test Lines Of Code (TLOC): This metric defines the cumulative number of lines of code of the JUnit test cases related to a use case (all of its scenarios). This number includes only the test cases related to classes/methods involved in the realization of the use case.

We conducted an experimental study using data collected from four open source Java projects. The selected case studies are from different domains and developed by different teams. Moreover, knowing that the OCP method is applied on UML class diagrams, we deliberately chose case studies that have been developed using (strictly) the object paradigm (in order to avoid biasing the OCP results). The use case models have been collected for each project by reverse engineering the source code. The goal was, because the use case models were not available, to produce (or reproduce) these models (and particularly the system sequence diagrams) as accurately as possible (without alteration) from code analysis. In fact, we have used the source code of the projects as well as the available documentation (particularly the detailed user guides).

Moreover, as the OCP method requires metrics that are computed from the class diagram, we also reverse engineered the source code of the selected systems to obtain class diagrams. In order to compare the two methods on the same basis, and knowing that the use case metrics are computed for each use case, we considered for the calculation of the OCP values all the classes (and methods) required for the implementation of the use case. So, for the calculation of the OCP values, we considered for each use case the corresponding part of the class diagram. Moreover, for the two first case studies, we developed the corresponding JUnit test cases. For the two other case studies, we used the JUnit test cases that were available on their respective web sites. For each system, we related to each use case the corresponding source code and JUnit test cases. For each use case, we calculated the values of the use case metrics and the global value of the OCP method. We also used the metric SLOC to quantify the source code corresponding to each use case, and the metric TLOC to quantify the JUnit test cases corresponding to each use case. In order to compare empirically the two methods, in terms of predicting SLOC and TLOC, we used both simple and multiple linear regression methods to build the prediction models.

Table 1: Descriptive statistics.

Variables	Obs	Min	Max	Mean	SD
NIC	33	1,000	15,000	4,424	3,307
NEO	33	1,000	36,000	3,333	6,352
NS	33	1,000	32,000	3,455	5,391
NT	33	1,000	67,000	5,818	11,359
OCP	33	2,000	134,000	21,485	23,630
SLOC	33	4,000	399,000	66,061	76,652
TLOC	33	11,000	777,000	130,818	206,393

The selected projects are: ATM, NextGen, CommonsEmail and JODA-Time. The first case study ATM[1] is a simulator system allowing performing basic banking operations (withdrawal, deposit, transfer, balance, etc.). We adapted the case study for our purposes. The second case study NextGen is an extension of the application developed by Larman in (Larman, 2004). The original application has been extended for our purposes by adding features about accounts receivable management, suppliers, and employees. We also added features to support billing and rental payments by debit and credit. The third case study Commons Email[2] aims to provide an API for sending email. It is built on top of the Java Mail API, which it aims to simplify. The fourth case study JODA-Time[3] is the de facto standard library for advanced date and time in Java. It provides a quality replacement for the Java date and time

[1] http://www.math-cs.gordon.edu/courses/cs211/ATMExample/
[2] https://commons.apache.org/proper/commons-email/
[3] http://www.joda.org/joda-time/

classes. Because the main data set was heterogeneous, and in order to have a significant sample of data, we decided to perform our study on the whole data set obtained by grouping the use cases of the four case studies. We have then a total of 33 use cases. Table 1 lists the descriptive statistics for the use case metrics, the global value of OCP and the size metrics SLOC and TLOC.

4.2 Correlation Analysis

In order to investigate the relationships between use case metrics (noted UCM_i), including the OCP global value, and the size metrics SLOC and TLOC, we performed statistical tests using correlation. The null and alternative hypotheses that our study has tested were:

H0: There is no significant correlation between a use case metric UCM_i (OCP) and SLOC (TLOC).

H1: There is a significant correlation between a use case metric UCM_i (OCP) and SLOC (TLOC).

For the analysis of the collected data, and in order to test the correlation between a use case metric (and OCP) and SLOC (TLOC), we used two correlation analysis techniques: Pearson and Spearman techniques. We used these techniques mainly for completeness. The Pearson r correlation is widely used in statistics to measure the degree of the relationship between linear related variables. The variables should be normally distributed. The Spearman rank correlation is a non-parametric test that is used to measure the degree of association between two variables. Spearman rank correlation test does not assume anything about the distribution of the variables.

Correlation is a bivariate analysis that measures the strengths of association between two variables. In statistics, the value of the correlation coefficient varies between +1 and -1. A positive correlation is one in which the variables increase together. A negative correlation is one in which one variable increases as the other variable decreases. A correlation of +1 or -1 will arise if the relationship between the variables is exactly linear. A correlation close to zero means that there is no linear relationship between the variables.

We used the XLSTAT[4] tool to measure the two types of correlations. We applied the typical significance threshold (alpha = 0.05) to decide whether the correlations where significant. For each pair < UCM_i, SLOC (TLOC)>, we analyzed the

Table 2: Pearson's correlation values between the use case metrics (and OCP) and SLOC (and TLOC).

Variables	NIC	NEO	NS	NT	OCP	SLOC	TLOC
NIC	**1**	**0,494**	**0,650**	**0,583**	**0,815**	**0,668**	0,170
NEO	**0,494**	**1**	**0,872**	**0,972**	**0,824**	**0,772**	**0,790**
NS	**0,650**	**0,872**	**1**	**0,962**	**0,907**	**0,850**	**0,528**
NT	**0,583**	**0,972**	**0,962**	**1**	**0,893**	**0,841**	**0,695**
OCP	**0,815**	**0,824**	**0,907**	**0,893**	**1**	**0,937**	**0,577**

Table 3: Spearman's correlation values between the use case metrics (and OCP) and SLOC (and TLOC).

Variables	NIC	NEO	NS	NT	OCP	SLOC	TLOC
NIC	**1**	0,324	**0,512**	**0,402**	**0,794**	**0,673**	0,190
NEO	0,324	**1**	0,243	**0,759**	**0,477**	**0,582**	**0,365**
NS	**0,512**	0,243	**1**	**0,705**	**0,686**	**0,757**	**0,385**
NT	**0,402**	**0,759**	**0,705**	**1**	**0,703**	**0,820**	**0,612**
OCP	**0,794**	**0,477**	**0,686**	**0,703**	**1**	**0,923**	**0,497**

collected data set by calculating the (Pearson's and Spearman's) correlation coefficients. Table 2 and Table 3 summarize the results (respectively Pearson's and Spearman's correlation coefficients). The correlation coefficients that are significant (alpha = 0.05) are set in boldface in the two tables.

The first global observation that we can make from Table 2 and Table 3 is that there is a significant relationship (the correlation values are in bold face) between all the use case metrics, including the OCP global value, and the size metrics SLOC and TLOC, except for the pair (NIC, TLOC) in the case of the two techniques. According to the obtained results, we can therefore reasonably reject the null hypothesis H0. The other global observations that we can make from these tables are that:

- According to the Pearson technique, the use case metrics (including OCP) that are the most correlated to SLOC are respectively OCP (0.937), NS (0.850) and NT (0.841), and the use case metrics (including OCP) that are the most correlated to TLOC are respectively NEO (0.790), NT (0.695), which are both much better correlated to TLOC than the OCP metric (0.577).
- According to the Spearman technique, the use case metrics (including the OCP value) that are the most correlated to SLOC are respectively OCP (0.923), NT (0.820) and NS (0.757), and the use case metric that is the most correlated to TLOC is NT (0.612), which is much better correlated to TLOC than the OCP metric (0.497).

We can also see from Table 2 and Table 3 that

[4] http://www.xlstat.com/

the correlation measures are positive. This means that one variable (a use case metric (OCP)) increases as the other variable (SLOC, TLOC) increases. Overall, results suggest that the use case metrics (and OCP) can be used to predict SLOC (TLOC). This must, however, be validated.

4.3 Source and Test Code Size Prediction

We present, in this section, the different steps of the empirical study we conducted in order to compare the two approaches, UCM and OCP, in terms of predicting SLOC and TLOC. The goal here is to evaluate how accurately the two methods predict the size metrics SLOC and TLOC. We used different modeling techniques to build the prediction models: (1) we used the simple linear regression method to evaluate the individual effect of each use case metric (and OCP) on the size metrics SLOC and TLOC, identifying which metrics are significantly related to SLOC and TLOC, and (2) we used the multiple linear regression method to explore the combined effect of the use case metrics on SLOC and TLOC, investigating if the combination of the use case metrics will improve the accuracy of the prediction of the size metrics SLOC and TLOC.

Linear regression is a commonly used statistical technique that is widely used to model (linearly approximate) the relationship between a dependent variable *y* and one or more explanatory variables denoted *X*. Linear regression analysis is of two types: The case of one explanatory variable is called *simple linear regression* (SLR). For more than one explanatory variable, it is called *multiple linear regression* (MLR). The *simple linear regression* analysis is based on the equation:

$$Y = \beta_0 + \beta_1 X, \qquad (1)$$

where Y is the dependent variable (SLOC or TLOC) and X is the independent variable (use case metric or OCP). The *multiple linear regression* analysis is based on the equation:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n, \qquad (2)$$

where Y is the dependent variable (SLOC or TLOC) and the X_i are the independent variables (use case metrics).

4.3.1 Hypotheses

The study tested various hypotheses, which relate the use case metrics (including OCP) to SLOC (TLOC). For each use case metric UCM_i:

The null hypothesis was: A use case with a high UCM_i (OCP) value is no more likely to have a high value of SLOC (TLOC) than a use case with a low UCM_i (OCP) value.

The hypothesis was: A use case with a high UCM_i (OCP) value is more likely to have a high value of SLOC (TLOC) than a use case with a low UCM_i (OCP) value.

4.3.2 Simple Linear Regression Analysis

In the SLR analysis, and therefore in each SLR model, we considered only one predictor variable. The SLR models for predicting SLOC (and TLOC) based on the value of a use case metric (or OCP) have the form of the equation (1) presented in the introduction of section 4.3. We used the XLSTAT tool to perform the simple linear regression analysis and derive the simple linear regression models.

Table 4 summarizes the results of the SLR analysis for predicting SLOC. From Table 4, it can be seen that all the SLR models are significant (according to the corresponding p-value). The performances of the models vary, however, from one model to the other. From Table 4, we can see that the SLR model based on the metric OCP is the most accurate in terms of SLOC prediction (the corresponding R^2 value is 87.9%), followed by the SLR model based on the metric NS (the corresponding R^2 value is 72.3%), and in a third position the SLR model based on the metric NT (the corresponding R^2 value is 70.7%).

Table 5 summarizes the results of the SLR analysis for predicting TLOC. From Table 5, it can be seen that all the SLR models are here also significant (according to the corresponding p-value), except the one based on the use case metric NIC. The performances of the models vary here also from one model to the other. From Table 5, we can see that the SLR model based on the metric NEO is the most accurate in terms of TLOC prediction (the corresponding R^2 value is 62.4%), followed by the SLR model based on the metric NT (the corresponding R^2 value is 48.3%), and in a

Table 4: SLR analysis – Predicting SLOC - Results.

	NIC	NEO	NS	NT	OCP
R^2	44.6%	59.7%	72.3%	70.7%	87.9%
b	0.668	0.772	0.850	0.841	0.937
p-value	< 0.0001	<0.0001	<0.0001	< 0.0001	< 0.0001

third position the SLR model based on the metric OCP (the corresponding R^2 value is 33.3%)

Table 5: SLR analysis – Predicting TLOC - Results.

	NIC	NEO	NS	NT	OCP
R^2	2.9%	62.4%	27.8%	48.3%	33.3%
b	0.170	0.790	0.528	0.695	0.577
p-value	0.343	<0.0001	0.002	< 0.0001	0.000

4.3.3 Multiple Linear Regression Analysis

The main purpose of the MLR analysis here is to explore the potential of the combined effect of the use case metrics (as predictor variables) on the size metrics SLOC and TLOC (as dependent variables). The question here is whether the combination of the use case metrics will improve the accuracy of SLOC (TLOC) prediction. The MLR model for predicting SLOC (TLOC) based on the values of the use case metrics has the form of the equation (2) presented in the introduction of section 4.3. We used the XLSTAT tool to perform the multiple linear regression analysis and derive the multiple linear regression model based on the combination of the use case metrics.

Table 6 gives the results of the MLR analysis for the prediction of SLOC. From Table 6, it can be seen that the MLR model based on the use case metrics is significant (according to the corresponding p-value). From Table 6 and Table 4, we can see that the MLR model is most accurate in terms of SLOC prediction (the corresponding R^2 value is 93.6%) than all the SLR models based on each use case metric, including the one based on the OCP metric. So, it is clear that, when the use case metrics are combined, the prediction quality visibly increased.

Table 7 gives the results of the MLR analysis for the prediction of TLOC. From Table 7, it can be seen that here also the MLR model based on the use case metrics is significant (according to the corresponding p-value). From Table 7 and Table 5, we can see that here also the MLR model based on the use case metrics is most accurate in terms of TLOC prediction (the corresponding R^2 value is 81.0%) than all the SLR models based on each use case metric, including the one based on the OCP metric. So, it is clear here also that, when the use case metrics are combined, the prediction quality visibly increased.

Table 6: MLR analysis – Predicting SLOC - Results.

	NIC, NEO, NS, NT
R2	93.6%
p-value	< 0.0001

Table 7: MLR analysis – Predicting TLOC - Results.

	NIC, NEO, NS, NT
R2	81.0%
p-value	< 0.0001

4.4 Threats to Validity

The study presented in this paper should be replicated using many other OO software systems in order to draw more general conclusions. Indeed, there are a number of limitations that may affect the results of the study or limit their interpretation and generalization.

The achieved results are based on the data set we collected from only four open source Java projects. To perform our study, we grouped the use cases of the four case studies to build our data set. Even if the collected data set is statistically significant, we do not claim that our results can be generalized.

The use case models and class diagrams have been collected by reverse engineering the source code of the applications. In fact, we have used the source code of the projects as well as the available documentation (particularly the detailed user guides). By analyzing both source code and the documentation of the projects, we were able to reproduce the models (particularly the use case models) needed for our study. The collection of the models has been performed by the same person (third author of the paper). To reduce the effect of this threat, results were discussed, checked and validated by all the authors.

The complexity of use cases is determined in part by the number of transactions. The simplest way to count the number of transactions was to count the number of events included in the flow of events. The number of events may be affected by the style adopted (by developers during the analysis phase) in the design of use cases (use case model). So, it is important to clarify that a key prerequisite to applying the proposed approach (this is also valid for the other use cases based prediction methods) is that use cases of the target system have been identified (structured and documented) at a suitable level of detail, where each transaction is specified.

Furthermore, the MLR analysis has been performed using all the use case metrics. The goal was especially to explore (in comparison to the SLR models, particularly the one based on the OCP metric) the combined effect of the use case metrics on the size metrics. The fact that the use case metrics (according to the data collected) are in some cases highly correlated (correlation analysis) suggest that there is a redundancy in the information captured by the use case metrics (multicollinearity between the use case metrics as predictor variables), which may lead to an over fitting of the multiple regression model. The simplest way to resolve this problem is to reduce the number of use case metrics (collinear predictor variables) to a subset of independent predictor variables. In fact, the goal here was not to build the best multiple regression model based on a subset of independent use case metrics. So, the findings in this paper should be viewed as exploratory and indicative rather than conclusive. Results show at least that use case metrics offer a potential and simple way that can be used in early stages of the software development lifecycle to predict the source code and test code size. Moreover, the study has been performed on simple case studies. It is necessary to replicate the study on large systems.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we focused on the early prediction of both SLOC and TLOC for OO software development. The study aimed basically at comparing empirically the Use Case Metrics (UCM) method, which is a use case model based prediction method that we proposed in previous work, and the Objective Class Points (OCP) method in terms of early prediction of SLOC and TLOC. We used both simple and multiple linear regression methods to build the prediction models. We conducted an empirical comparison using data collected from four open source Java projects.

In a first step, we investigated the relationships between use case metrics (including OCP) and size metrics SLOC and TLOC. The goal here was to evaluate the capacity of use case metrics (and OCP) to predict SLOC and TLOC. In a second step, we used the simple linear regression method to evaluate the individual effect of each use case metric (including OCP) on SLOC (TLOC), and the multiple linear regression method to explore the combined effect of use case metrics on SLOC (TLOC).

In order to assess the relationships between the use case metrics, including the OCP global value, and SLOC (TLOC), we performed statistical tests using correlation. We used two correlation analysis techniques: Pearson and Spearman techniques. We observed a significant and (in some cases) a high correlation between the use case metrics (and OCP) and SLOC (TLOC). These correlations suggest that use case metrics (and OCP) can be used to predict the final source code (test code) size.

We also derived several simple linear regression models to relate SLOC (TLOC) to the use case metrics (and OCP), and a multiple linear regression model based on the use case metrics to predict SLOC (TLOC). Overall, results provide evidence that the multiple linear regression model, based on the combination of the use case metrics, is more accurate in terms of early prediction of SLOC and TLOC than: (1) the simple linear regression models based on each use case metric, and (2) the simple linear regression model based on the OCP metric. The combination of the use case metrics highly increased the accuracy of SLOC (TLOC) prediction. The main limitations of the observations made in this study are basically related to the small size (and number) of the considered set of projects.

The performed study should be replicated using many other OO software systems in order to draw more general conclusions. The findings in this paper should be viewed, in fact, as exploratory and indicative rather than conclusive. As future work, we plan to extend the present study by using other methods (machine learning methods particularly) to explore the individual and combined effect of the use case metrics on the source code (test code) size (and software development effort), compare our approach to other development effort prediction approaches, and finally replicate the study on various OO software systems to be able to give generalized results.

ACKNOWLEDGMENTS

This work was (partially) supported by NSERC (Natural Sciences and Engineering Research Council of Canada) grant.

REFERENCES

Abrahao, S., Poels, G., Pastor, O.: A Function Size Measurement Method for Object-Oriented Conceptual Schemas: Design and Evaluation Issues, Software and System Modeling, 5(1), 2006. *lokan, c.j.: Function Points, Advances in Computers*, 65, 2005.

Albrecht, A.: Measuring Application Development Productivity*, in IBM Application Development Symposium,* 1979.

Albrecht, A.J., Gaffney, J.E.: Software function, Source Lines of Code and Development effort Prediction: a Software Science Validation, *IEEE Transactions on Software Engineering, 9 (6),* 1983.

Antoniol, G., Lokan, C., Caldiera, G., Fiutem, R.: A Function Point-Like Measure for Object-Oriented software, *Empirical Software Engineering, 4 (3),* 1999.

Antoniol, G., Lokan, C., Fiutem, R.: Object-Oriented Function Points: An Empirical Validation*, Empirical Software Engineering, 8 (3),* 2003.

Badri, M., Badri, L., Flageol, W.: Predicting the Size of Test Suites from Use Cases: An Empirical exploration, H. Yenigün, C. Yilmaz, and A. Ulrich (Eds.): *ICTSS 2013, LNCS 8254, November, 2013*.

Badri, M., Badri, L., Flageol, W.: Simplifying Test Suites Size Prediction Using Use Case Metrics: An empirical comparison, Submitted to the International *Journal of Software Engineering and Knowledge Engineering (IJSEKE),* September 2015.

Badri, M., Toure, F.: Empirical Analysis of Object-Oriented Design Metrics for Predicting Unit Testing Effort of Classes, *Journal of Software Engineering and Applications, 5(7),* July 2012.

Bianco, V.D, Lavazza, L.: An Assessment of Function point-like metrics for object-oriented Open-Source Software, *International Conference on Software Process and Product Measurement,* 2006.

Bianco, V.D, Lavazza, L.: Object-Oriented Model Size Measurement: Experiences and a Proposal for a Process, Workshop on Model Size Metrics, *ACM-IEEE International Conference on Model Driven Engineering Languages and Systems*, 2006.

Boehm, B., Abts, C., Brown, A.W. et al.: *Software Cost Estimation with COCOMO II,* Prentice-Hall, Englewood Cliffs, NJ, 2000.

Boehm, B., Abts, C., Chulani, S.: Software Development Cost Estimation Approaches – *A survey, Annals of Software Engineering, 10*, 2000.

Boehm, W.B.: *Software Engineering Economics*, Prentice-Hall, 1981.

Bourque, P., Côté, V.: An experiment in software sizing structured analysis metrics, *Journal of Systems and Software, 15 (2),* 1991.

Bruntink, M., Van Deursen, A.: Predicting Class Testability Using Object-Oriented Metrics. In: *Proceedings of the 4th IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2004)*, pp. 136–145, September 2004.

Bruntink, M., Van Deursen, A.: An Empirical Study into class testability, *Journal of Systems and Software, 79(9),* 1219–1232, 2006.

Carbone, M., Santucci, G.: Fast & Serious: A UML based Metric for Effort Estimation, *6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, 2002.

Chen, Y., Boehm, B.W., Madachy, R., Valerdi, R.: An EMPIRICAL Study of eServices Product UML Sizing Metrics, *International Symposium on Empirical Software Engineering*, 2004.

Carroll, E.R.: Estimating Software Based on Use case Points, *OOPSLA'05*, San Diego, California, USA, October 16-20, 2005.

Costagliola, G., Ferruci, F., Tortora, G., Vitiello, G.: Class Point: an Approach for the Size Estimation of Object-Oriented Systems, *IEEE Transactions on Software Engineering, 31(1)*, 2005.

Hastings, T.E., Sajeev, A.S.M.: A vector-Based Approach to Software Size Measurement and Effort Estimation, *IEEE Transactions on Software Engineering, 27* (4), 2001.

Henderson-Sellers, B.: Corrigenda: Software Size Estimation of Object-Oriented Systems, *IEEE Transactions on Software Engineering, 23 (4),* 1997.

Jacobson, I., Christerson, M., Jonson, P., Overgaard, G.: Object-Oriented Software Engineering: *A Use Case Driven Approach,* Addison-Wesley, 1993.

Jahan, M.V., Sheibani, R.: A new Method for Software Size Estimation based on UML Metrics*, The First Conference on Software Engineering*, Islamic Azad University, 2005.

Janaki Ram, D., Raju, S.V.G.K.: Object-Oriented Design Function Points, *1st Asia-Pacific Conference on Software Quality*, 2000.

Jorgensen, M., Shepperd, M.: A systematic Review of Software Development Cost Estimation Studies, *IEEE Transactions on Software Engineering, 33 (1),* 2007.

Jorgensen, M., Boehm, B.: Software Development Effort Estimation: Formal Methods or Expert Judgment?, *IEEE Software, 2009*.

Kemerer, C.F.: An empirical validation of software cost estimation, *Communications of the ACM*, 30 (5), 1987.

Kim, S., Lively, W., Simmons, D.: An effort Estimation by UML Points in the Early Stage of Software development, *International Conference on Software Engineering Research & Practice*, 2006.

Lagerstrom, R., Marcks von Wurtemberg, L., Holm, H., Luczak, O.: Identifying factors affecting software development cost and productivity*, Software Quality Journal*, 2012.

Larman, C.: Applying UML and Design Patterns, An *Introduction to Object-Oriented Analysis and Design and the Unified Process*, Prentice Hall, 2004.

Leung, H., Fan, Z.: Software Cost Estimation, Handbook of Software Engineering and Knowledge Engineering, Vol. 2, *World Scientific Publishing, 2002.*

Matson, J.E., Barrett, B.E., Mellichamp, J.M.: Software Development Cost Estimation using Function Points, *IEEE Transactions on Software Engineering, 20 (4),* 1994.

McDonell, S.G.: Software Source Code Sizing using Fuzzy Logic Modeling, Information and Software Technology, 45 (7), 2003.

Mishra, S., Tripathy, K.C., Mishra, M.K.: Effort Estimation based on Complexity and Size of relational database system, *International Journal of Computer Science & Communication, 1 (2),* 2010.

Ochodek, M., Nawrocki, J., Kwarciak, K.: Simplifying effort estimation based on Use Case Points, *Information and Software Technology, 53, 200–213,* 2011.

Pfleeger, S.L., Wu, F., Lewis, R.: Software cost estimation and sizing methods: Issues and Guidelines, *RAND Corporation, 2005.*

Singh, Y., Kaur, A., Malhota, R.: Predicting Testability effort using Artificial Neural Network, *In: Proceedings of the World Congress on Engineering and Computer Science, WCECS,* San Francisco, USA, 2008.

Singh, Y., Saha, A.: Predicting Testability of Eclipse: a Case Study, *Journal of Software Engineering, 4(2),* 2010.

Trendowicz, A., Munch, J., Jeffery, R.: State of the Practice in Software Effort Estimation: A survey and Literature Review, *CEE-SET 2008, LNCS 4980, 2008.*

Verner, J., Tate, G.: A software size model, *IEEE Transactions on Software Engineering,* 38 (5), 2012.

Zhao, Y., Tan, H.B.K.: Software Cost Estimation through Conceptual Requirement*, 3rd International Conference on Software Quality*, 2003.

Zhou, Y., Leung, H., Song, Q., Zhao, J., Lu, H., Chen, L. and Xu, B.: An In-Depth Investigation into the Relationships between Structural Metrics and Unit Testability in OOS, *Information SCIENCES, 55(12)* Science China, 2012.

Zhou, Y., Yang, Y., Xu, B., Leung, H., Zhou, X.: Souce Code Size Estimation Approaches for Object-Oriented systems from UML Class Diagrams: A Comparative study, *Information and Software Technology, 56,* 2014.