



# International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

(Open Access, Double Blind Peer-reviewed, Refereed and Indexed Journal)

[www.iasir.net](http://www.iasir.net)

## An Integrated Metric for Software Change Prediction

Ashu Jain

Northern India Engineering College  
Guru Gobind Singh Indraprastha University, Dwarka  
New Delhi, India

**Abstract:** Software maintenance is the most expensive and time consuming phase of software development life cycle (SDLC). The prediction of the software changes in the early phases of SDLC can direct us to use the resources efficiently, which can thus help in saving the time and cost associated in maintaining the software. In this paper, two techniques Feature Subset Selection and Bivariate correlation analysis for building the software maintainability model have been evaluated. A combination of Static and Dynamic metrics has been used to develop the models. Dynamic metrics unlike static metrics also consider the runtime behavior of the software. An open source software jTDS has been used for developing the model. Results shows that the model developed using the feature subset selection technique with a correlation coefficient 0.603 performs better than the model developed using Bivariate correlation analysis technique with a correlation coefficient 0.475. Based on the results, a new metric Software Maintainability Factor (SMF) has been proposed for predicting the change in terms of static and dynamic metrics, in the early phases of SDLC. Two other open source software – jXLS and orDrumbox have been used for validating the SMF metric.

**Keywords:** software maintenance, static metrics, dynamic metrics, feature subset selection, change prediction, validation

### I. Introduction

Software maintenance is the changes made to the software in the form of addition, deletion, or modification of source code lines. It is the last phase of software development life cycle (SDLC) and incurs the maximum time and cost. On the other hand, software maintainability [1] is the quality attribute of the software, which can be defined as the ease with which these changes are performed into the software. The prediction of these changes in the early phases of the SDLC can help the software developers to invest their resources in the required parts of the software, which in turn will result in saving time and money. Change can be predicted in terms of internal attributes of the software which are available in the early phases of SDLC. It can be seen from literature [2,3,4,5,6] that there exists a strong relationship between software metrics and software maintainability.

In this study, we have used a combination of static and dynamic software metrics [7] as internal attributes, i.e. change has been predicted in terms of the values of static and dynamic software metrics.

Static software metrics consider only the structural behavior of the software. They are computable in the early phases of SDLC. No real time execution need take place for the collection of static metrics. The basic purpose of using the static metrics is to state about the size and complexity of the software. The first static metric proposed was source lines of code (LOC) [8].

Dynamic metrics are the software metrics which also considers the runtime behavior of the software along with the structural behavior. They take into consideration the time factor which is neglected by static metrics. Dynamic metrics give more accurate results in comparison to static metrics, for software maintainability prediction.

To create the model, we have used feature subset selection (FSS) method and Bivariate Correlation analysis. FSS technique is used to reduce the number of variables for constructing model. It selects the significant variable predictors out of many variables. On the other hand, Bivariate correlation analysis shows the linear correlation between two variables. Its value varies between -1 and +1. -1 means total negative correlation and +1 means total positive correlation. In this paper, an open source software jTDS has been used for constructing the models. Based on the results, a new metric Software Maintainability Factor (SMF) has been proposed which can predict the number of changes to be required in each class of the software. To validate the SMF, other two open source software – jXLS and orDrumbox, have been used.

The rest of the paper has been organized in the following manner: section II presents the related work which has been done in this field, section III describes the empirical data collection, section IV describes the prediction models generated in the study, section V explains the new metric being proposed, section VI presents the validation measures and validation results of the proposed metric, section VII describes the application of the proposed metric, section VIII describes the limitations of the study, and finally section IX explains the conclusion and future scope of the conducted study.

## II. Related Work

Few studies which have been done in the field of change prediction have been summarized here. Malhotra and Khanna [9] compared the performance of various machine learning and search based techniques for predicting the change in software. Hayes [10] et al presented the model for estimating adaptive software maintenance effort in person hours, the Adaptive Maintenance Effort Model (AMEffMo). Ye et al [11] proposed the Software Maintainability Evaluation Model based on Multiple Classifiers Combination (SMEM-MCC) which is a software metrics based evaluation method. They showed that SMEC-MCC work better than any other single classifier, such as BPNN, SMO or decision tree. Manchanda and Chug [12] used the feature subset selection (FSS) to reduce the number of metrics for software change prediction. They analyzed the combination of FSS techniques and Machine learning techniques for getting the best results. Gupta and Chabra [13] compared the performance of various approaches for the dynamic analysis of object oriented systems. He concluded that Aspect Oriented Programming (AOP) is the most efficient technique for the dynamic analysis and easier to implement. Dagpinar and Jhanke [14] investigated the object oriented metrics which can be used as the significant predictors for software maintainability. Results indicated that size and import direct coupling metrics are significant predictors of software maintainability. Aggarwal et al [15] proposed the four parameter integrated measure of software maintainability using a fuzzy model. They used data of maintenance time of projects to validate the proposed model. Elish and Elish [16] empirically investigated the TreeNet model for software maintainability prediction. Sharma and Chug [7] validated the superiority of dynamic metrics over static metrics, for software maintainability prediction, with the help of four machine learning algorithms. Chhabra and Gupta [17] carried out the survey of the existing dynamic metrics and also presented the advantages of dynamic metrics over static metrics. Although many authors used the static metrics and dynamic metrics for constructing the software maintainability model, and were successful, but none of them proposed the use of the combination of static and dynamic metrics for developing the software maintainability model. In this study, a combination of static and dynamic metrics has been used to create the two software maintainability models, and based on the results obtained, a new metrics software maintainability factor (SMF) has also been proposed, which can predict the number of changes need to be performed in each class of the software.

## III. Empirical Data Collection

### A. Dependent and Independent variable

Change has been taken as the dependent variable which is calculated by measuring the number of source code of lines which have been added, deleted or modified, from one version of software to another. Addition and deletion of source code of line is counted as one change, whereas modification of a source code of line is counted as two changes. This procedure is followed for each class of the software.

A combination of static and dynamic software metrics is used as the independent variables which have been described in table I and table II respectively. Static metrics are computed using the ckjm tool [18]. Dynamic metrics are computed by using the Netbeans plug-in Source Code metrics [19].

### B. Open Source Software

In order to perform this study, three open source software downloaded from sourceforge.net, have been used. To create the model using FSS and Bivariate Correlation analysis technique, jTDS (version 1.2.8 and 1.2.9) open source software has been used. It is an open source jdbc 3.0 type 4 driver which is used for Microsoft SQL Server. It contains 64 classes. The proposed metric is validated using the other two open source software – orDrumbox and jXLS. OrDrumBox (version 0.9.08 and 0.9.081) is a Software Drum Machine and Audio Sequencer in Java. It is used for Creating songs with the use of Drum Kits. it also has creative automatic compositions features, step sequencer, polyrythmes, drumkit manager, midi import/export, wav import/export, soft synth, piano roll, arpeggiators. It contains 218 classes. JXLS (version 1.0.5 and 1.0.6) is a small and easy-to-use Java library for writing Excel files using XLS templates and reading data from Excel into Java objects using XML configuration. It contains 78 classes.

## IV. Prediction Models

### A. FSS based Model

Feature subset selection is the process of finding the subset of significant variables which can serve as predictors for generating the prediction model. This process is used to reduce the number of variables, which in turn reduces the time for model construction. This process works by eliminating those features which are either irrelevant or redundant.

The basic algorithm used for FSS is following:

Algorithm for FSS:

Given: A set of predictors (dynamic software metrics) P and a target variable (Change) T.

Find: A set S, minimum subset of P such that it achieves maximum accuracy for classifying the T.

In this study, CFS based FSS algorithm [12] is used to select the significant predictor metrics. The metrics that have been chosen by it are DIT, NOSF, NOSM, TCC and WMC. After the selection of these metrics, equal weightage of 0.2 is given to all metrics for generating the prediction model.

$$\text{Change} = 0.2 * \text{DIT} + 0.2 * \text{NOSF} + 0.2 * \text{NOSM} + 0.2 * \text{TCC} + 0.2 * \text{WMC} \quad (1)$$

Table I. STATIC METRICS

| Metric                             | Description                                                                                                                                               |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Weighted Methods per Class (WMC)   | Number of methods in a class. Low value of WMC is required. High value leads to more faults.                                                              |
| Depth of Inheritance Tree (DIT)    | Maximum length of path from a particular class to root node. Low value is required. High value of DIT increases number of faults.                         |
| Number of Children (NOC)           | Number of immediate derived classes of a particular class. High NOC is recommended. High value reduces the number of faults.                              |
| Coupling between Objects(CBO)      | Count of number of classes to which given class is coupled. High value of CBO is undesirable since it leads to difficulty in maintenance.                 |
| Response for Class(RFC)            | Number of methods that get executed on receiving a particular message from an object of a class. High value of RFC results in more faults.                |
| Lack of Cohesion of Methods (LCOM) | Number of pair of methods that access disjoint variables minus the number of pair of methods that access the same variable. Low value of LCOM is desired. |

Table II. DYNAMIC METRICS

| Metric                              | Description                                                                           |
|-------------------------------------|---------------------------------------------------------------------------------------|
| Loose Class Coupling (LCC)          | Measures the low dependency of a class on other classes at run time.                  |
| Tight Class Coupling (TCC)          | Measures the high dependency of a class on other classes at run time.                 |
| Lack of cohesion of methods (LCOM)  | Calculates the lack of cohesion of methods at run time low value of LCOM is required. |
| Lines of Code (LOC)                 | Number of lines of code executed at run time                                          |
| Lines of Comment (LOCm)             | Number of lines of comments used at run time.                                         |
| Number of Assertions per KLOC (NAK) | Number of declarations per KLOC at execution time.                                    |
| Number of Children (NOC)            | Number of immediate subclasses of a class at run time.                                |
| Number of Fields (NOF)              | Total number of fields defined at run time                                            |
| Number of Methods (NOM)             | Total number of methods defined at run time.                                          |
| Number of Static Fields (NOSF)      | Total number of static fields defined at run time.                                    |
| Number of Static Methods (NOSM)     | Total number of static methods defined at run time.                                   |
| Number of Test Methods (NTM)        | Total number of test methods which get executed.                                      |
| Weighted Method Count(WMC)          | Calculates method complexity at run time.                                             |

### B. Bivariate correlation based model

Bivariate correlation measures the degree of relationship between two variables. Its value can range from absolute value 1 to 0. If its value is close to +1, then it signifies the strong positive relationship. The relationship between two variables can be positive or negative. If the value of a variable increases if we increase the value of another variable, then it is positive relation between those variables. But, if the value of a variable decreases on increasing the value of another variable, then its is called as negative relation.

In this study, Correlation is measured between each metric and the dependent variable change. Only those metrics are considered for model generation, which have been found to be significantly correlated with the dependent variable change. The metrics which have been found to be significantly correlated are CBO, RFC, NOF, NOSF, NOSM, WMC and LOC. The value of correlation coefficient is taken as the weights for the metrics.

$$\text{Change} = 0.331 * \text{CBO} + 0.342 * \text{RFC} + 0.461 * \text{NOF} + 0.505 * \text{NOSF} + 0.438 * \text{NOSM} + 0.586 * \text{WMC} + 0.445 * \text{LOC} \quad (2)$$

After calculating the change using these prediction models, correlation coefficient is calculated between the actual and predicted change, which is found to be 0.603 for FSS and 0.475 for bivariate correlation. Since the value of correlation coefficient for FSS based model is more than the bivariate correlation based model, FSS based model can be used as the prediction model for software maintainability.

### V. Software Maintainability Factor

In this section a new metric, Software Maintainability factor has been described, which can be used for predicting the number of changes required to be made in each class of the software. The proposed metric has been further validated on two open source data set.

Based on the results of the jTDS data set using the FSS based technique, we propose a Software maintainability factor (SMF) metric which can be used for predicting the number of changes to be required

The metric definition is as follows:

For a class  $C_i$ , the Change can be calculated by using the following equation:

$$\text{SMF} = 0.2 * \text{DIT} + 0.2 * \text{NOSF} + 0.2 * \text{NOSM} + 0.2 * \text{TCC} + 0.2 * \text{WMC} \quad (3)$$

Value of SMF and number of changes are directly proportional to each other. That is, increase in value of SMF will increase the number of changes and vice-versa.

## VI. Proposed Metric validation

### A. Validation Measures

For the validation of the proposed metric, we have used the prevalent accuracy measures as proposed by Kitchenham [20], and correlation coefficient between the actual change and the predicted change is also calculated. The performance of the SMF metric is evaluated with the help of following three measures.

#### 1. Mean absolute Error

It is defined as the mean of absolute difference between the actual and predicted values of change. The mathematical formula for calculating it is given in equation (4).

$$MAE = \frac{\sum |f_i - y_i|}{n} \quad (4)$$

Where  $f_i$  is the actual value,  $y_i$  is the predicted value and  $n$  is total number of classes.

#### 2. Root Mean Squared Error

It is defined as the standard deviation between the actual and the predicted values of change. The mathematical formula for calculating it is given in equation (5).

$$RMSE = \sqrt{\frac{\sum (f_i - y_i)^2}{n}} \quad (5)$$

Where  $f_i$  is the actual value,  $y_i$  is the predicted value and  $n$  is total number of classes.

#### 3. Correlation coefficient

It tells about the linear correlation between the two variables. The value of coefficient ranges from -1 to +1. Here the Minus (-) sign depicts the negative correlation and plus (+) sign depicts the positive correlation between the two variables. It can be calculated by using the following formula:

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{n}}{\sqrt{\left(\sum X^2 - \frac{\sum X^2}{n}\right) \left(\sum Y^2 - \frac{\sum Y^2}{n}\right)}} \quad (6)$$

Where  $n$  is the number of classes,  $\sum X$  is the sum of the values of first variable,  $\sum Y$  is the sum of the values of second variable,  $\sum X^2$  is the sum of the squares of the values of first variable, and  $\sum Y^2$  is the sum of the squares of the values of second variable

Lower the value of MAE and RMSE, and, higher the value of correlation coefficient, higher the accuracy of the metric.

### B. Validation Results

In order to validate the proposed SMF metric, other two open source software – jXLS and orDrumbox have been used. The validation results of SMF metric have been summarized in table III. From table III, it can be seen that jXLS has the correlation coefficient 0.862 and orDrumBox has the correlation coefficient 0.614, which shows that there exists a significant relation between the actual change and predicted change. The value of MAE and RMSE for both the datasets is also low. Hence SMF can be used as the effective change prediction metric.

Table III. VALIDATION RESULTS OF SMF METRIC

| Dataset   | Correlation coefficient | Sig. (2-tailed) | MAE  | RMSE  |
|-----------|-------------------------|-----------------|------|-------|
| orDrumbox | 0.614                   | .000            | 5.12 | 12.36 |
| jXLS      | 0.862                   | .000            | 4.23 | 7.40  |

## VII. APPLICATION OF THE SMF METRIC

The SMF metric can be used to identify the number of changes in each class of the software, in the early phases of SDLC. This will help the managers in better software project management. Resources will be spent only on those classes which will go under change in the upcoming versions of the software. SMF value represents the number of changes required in the class, which can be due to the presence of bad smells. Hence, refactoring techniques will be applied only to those classes which show the high value of SMF. Testing effort will also be spent on classes with high value of SMF.

The SMF metric will help the software practitioners in developing the good quality software by spending in effort and resources only to the required classes of software.

## VIII. CONCLUSION AND FUTURE SCOPE.

In this study, the performance of FSS technique and Bivariate correlation analysis is evaluated using dynamic software metrics as independent variables for change prediction. For this purpose, jTDS software has been used. Based on the results obtained, a new metric Software Maintainability Factor (SMF) has been proposed, which can

identify the number of changes required in each class of the software. This metric is validated by using two other open source software – jXLS and orDrumbox. The conclusions have been summarized in the following points.

1. It is found that the correlation coefficient between the actual change and predicted change using FSS is 0.603 and 0.475 using Bivariate correlation technique. Hence FSS can be used as the change prediction model in the early phases of the SDLC.
2. The metrics DIT, NOSF, NOSM, TCC and WMC are selected as significant predictors for change using FSS.
3. A metric Software Maintainability Factor (SMF) is proposed for predicting change in early phases of SDLC. The validation results of SMF for open source software jXLS (Correlation Coefficient: 0.862, MAE: 4.23, RMSE: 7.40) and orDrumbox (Correlation Coefficient: 0.614, MAE: 5.12, RMSE: 12.36) show that SMF can be used effectively for software change prediction.

Our future work aims to generalize the results by replicating this study on larger datasets. We also plan to conduct our study on the software projects of different languages.

## IX. References

- [1] IEEE Std. 610.12-1990.: Standard Glossary of Software Engineering Terminology, IEEE Computer Society Press, Los Alamitos, CA, 1993
- [2] K.K. Aggarwal, Y. Singh, P. Chandra, and M. Puri, "Measurement of Software Maintainability Using a Fuzzy Model", *Journal of Computer Sciences*, vol. 1, no.4, pp. 538-542, 2005 ISSN 1549- 3636 © 2005 Science Publications.
- [3] M. Thwin, and T. Quah, "Application of neural networks for software quality prediction using object oriented metrics", *Journal of Systems and Software*, vol. 76, no. 2, pp. 147-156, 2005
- [4] S. Misra, "Modeling design/coding factors that drive maintainability of software systems", *Software Quality Journal*, vol. 13, no. 3, pp. 297-320, 2005.
- [5] C.V. Kotten, and A.R. Gray, "An application of Bayesian network for predicting object-oriented software maintainability", *Information and Software Technology Journal*, vol: 48, no : 1, pp 59-67, Jan 2006.
- [6] A. Chug, and A. Sawhney, "The Comparative Study of Software Maintenance Forecasting Analysis based on Backpropagation (NFTOOL and NTSTOOL) and Adaptive Neuro Fuzzy Inference System(ANFIS)", *International Journal of Advancements in Research & Technology*, Volume 2, Issue 5, May-2013.
- [7] H. Sharma,, and A. chug, "Dynamic Metrics are Superior than Static Metrics in Maintainability Prediction : An Empirical Case Study", 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO), pp. 1-6, 2015
- [8] L. Bennet and S. E. Burton, "Problems in application software maintenance", *Communications of the ACM*, vol. 24, no.11, pp. 763-769,1981
- [9] R. Malhotra, and M. Khanna, "Mining the impact of object oriented metricsfor change prdiction using Machine Learning and Search based techniques", 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 228-234,2015
- [10] J. H. Hayes, H.C. Patel, and L. Zhao, "A Metrics-Based Software Maintenance Effort Model ", *Eighth European Conference on Software Maintenance and Reengineering*, pp. 254-258,2004
- [11] F. Ye, X. Zhu, and Y. Wang, "A New Software Maintainability Evaluation Model Based on Multiple Classifiers Combination", *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)*, pp. 1588-1591, 2013
- [12] S. Manchanda and A. Chug, "CFS based Feature Subset Selection for Software Maintainance Prediction", *International Journal of Advance Foundation and Research in Computer (IAFRC)*, Vol. 2, Issue 5, ISSN 2348 – 4853, 2015
- [13] V.Gupta, and J. K. Chabra "Measurement of Dynamic Metrics Using Dynamic Analysis of Programs", *Applied Computing Conference*, Istanbul, Turkey, pp. 27-37, 2008.
- [14] M. Dagpinar, and J.H Jahnke, "Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison", *Proceedings of the 10th Working Conference on Reverse Engineering (WCRE'03)* 1095-1350/03, 2003
- [15] K.K. Aggarwal, Y. Singh, P. Chandra, and M. Puri, "Measurement of Software Maintainability Using a Fuzzy Model", *Journal of Computer Sciences* 1 (4): 538-542, ISSN 1549-3636, Science Publications, 2005
- [16] M.O. Elish, and K.O. Elish, "Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study", *European Conference on Software Maintenance and Reengineering*, pp. 1534-5351, 2009, DOI 10.1109/ CSMR.2009.57.
- [17] J. K. Chabra and V. Gupta, "A survey of dynamic software metrics", *Journal of Computer Science and Technology* 25(5): 1016–1029, DOI 10.1007/s11390-010-1080-9,2010
- [18] D. Spilellis, "Tool Writing: A forgotten Art?", *IEEE Software*, 22(4), doi:10.1109/MS.2005.111, 2005
- [19] <http://plugins.netbeans.org/plugin/42970/sourcecodemetrics>
- [20] B.A. Kitchenham, C.M. Pickard, S.G. MacDonell, and M.J. shepherd, "What accuracy statistics really measure", *Proceedings-Software, IEEE*, vol. 148, no. 3, pp. 81-85,2001.