International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 9 (2017) pp. 2038-2043

Software Fault-proneness Prediction using Module Severity Metrics

Euyseok Hong
Department of Information Systems Engineering, Sungshin Women's University, Seoul, Korea.
ORCID: 0000-0003-1882-0341

Abstract
Most of the fault prediction studies have focused on the binary classification models that determine whether the input modules are fault-prone or not. More recently, several studies have shown that severity-based multi-classification models are more useful since they can predict the fault-proneness depending on the severity of the defects in the module. We present new severity-based prediction models using two module severity metrics proposed in our previous study. Those metrics defined to measure the severity of a module are used to define the output values of the prediction model, i.e., the severity-based fault-proneness of a module. Experimental results show that the proposed MS model outperforms previous models in terms of AUC and Accuracy.

Keywords: Fault-proneness prediction, Module severity metrics, Defect severity

INTRODUCTION

Software defect is an imperfection or deficiency in a software product where that product does not meet its requirements or specifications and needs to be either repaired or replaced [1]. The metric-based software fault prediction model receives the quantified modules or classes as metric vectors and predicts their fault information. Most of the fault prediction studies have focused on the binary classifications that predict whether input modules are fault-prone or not [2-4]. The main problem of the model which simply predicts the presence or absence of the defects is that it does not consider the defect attributes such as severity or priority. Defect severity is a measure of the impact a defect has on a system and its users [5]. The ability to predict module fault-proneness in various severity categories such as high, low, and not fault-prone as shown in Figure 1, is much more useful than the binary classification due to the fact that not all defects have the same severity. Thus, the severity-based fault prediction model enables higher quality testing, refactoring, and resource allocation at a lower cost by predicting the critical trouble spots in the system.

There is no agreed-upon standard to define the defect severity value although IEEE Std. 1044-2009 specifies the examples of the severity level. For example, some projects in the NASA metrics data program (MDP) dataset have severity values at five levels, with severity 1 representing the most severe defects and severity 5 the least severe.

Previously proposed severity-based prediction models were all supervised learning models using the training data. Zhou and Leung [6] used logistic regression, Naïve Bayes, Random Forest, and NNge as classification algorithms and classified severity 1 defects as high severity defects and 2-5 defects as low severity defects. Singh et al. [7] used logistic regression, decision trees, and artificial neural networks as classification algorithms, and classified severity 1 defects as high severity defects, 2 as medium severity defects, and 2-5 defects as low severity defects. Both studies used KC1, an object-oriented project of NASA dataset, and showed that most of the C&K metrics [8], except DIT, are closely related to the severity-based fault-proneness of a class. Shatnawi and Li [9] performed a study on three release data from the Eclipse project to determine if the prediction model could determine the severity-based fault-prone classes of the next release. In our previous study, we used JM1 and PC4 projects of NASA data set to create a ternary classification model as shown in Figure 1, and the backpropagation neural network model showed the best prediction performance [10].

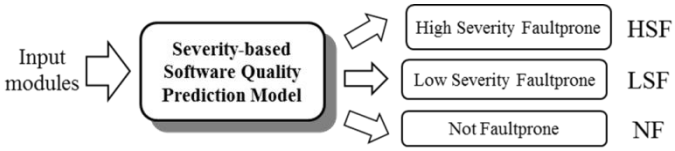

Figure 1. Ternary classification model structure

All previous studies used the severity of defects in a module to define the output values of the predictive model, that is, the severity-based fault-proneness of the module. More specifically, in Figure 1, a module that has one or more high severity defects is a high severity fault-prone (HSF) module; a module that does not have a high severity defect and has one or more low severity defects is a low severity fault-prone (LSF) module; and a module that does not have defects is not fault-prone (NF) module. However, this simple method does not take into account the number of defects in a module that can affect the severity of a module, and has difficulty in determining which severity level should be the high severity boundary.

In this paper, we propose a new method to define the severity-based fault-proneness of a module using the module severity metrics proposed in our previous study [11]. Another important objective of this paper is to verify the usefulness of the module severity metrics by comparing the prediction performance of the new prediction models using the proposed method with the previous models.

MODULE SEVERITY METRICS

The two main factors that affect a module severity are the number of defects the module has and the severity values of the defects. If all defects in a module have the same severity

value, the more defects in a module, the more severe the module. Also, if the number of defects that each module has is the same, a module with more severe defects is more severe.

We defined new severity metrics to measure module severity in the previous study. In this paper, we use the three metrics to construct prediction models. First, the maximum severity level of a module *m*, i.e., MSL_m, is defined as follows:

$$MSL_m = max_{i \in D_m} dsl_i$$

where D_m is a set of the defects in the module *m*, and dsl_i represents the defect severity level of the defect *i*. In most defect severity studies, the more severe the defect, the lower the severity level value. However, *dsl* starts at level 1 and gets higher as the defect become more severe. The *dsl* values of the defects in NASA data set are in the range of 1 to 5 and 5 means the most severe defect.

Second, the module severity of a module m, i.e., MS_m, is defined as follows:

$$MS_m = \sum_{i \in D_m} a^{dsl_i} \qquad s.t.\ a \geq 2$$

where a is an integer greater than or equal to 2. *dsl* representing the defect severity level is an ordinal scale metric that indicates only the order which defect is more severe. The severity differences between severity levels are not linear, and as the severity level increases, the value of the actual severity increases significantly beyond the linear relationship. Since these differences depend on the characteristics of the software development project, the module, and the defects, it is not possible to define the severity accurately using the severity level. However, as the severity level increases, it is certain that the severity will increase faster and larger than the linear relationship. Therefore, the severity of the defect is defined as an exponential expression, which is one of the common types with this property, and a is set to 2 in this paper.

Finally, the module severity density of a module m, i.e., MSD_m, is defined as follows:

$$MSD_m = \frac{MS_m}{SLOC} \times 100$$

MS is more likely to be larger for larger modules, but *MSD* which measures module severity per 100 SLOC is independent of module size.

MODEL CONSTRUCTION

Data Set

Among the two versions of the NASA dataset, we use the original NASA dataset for building models since the PROMISE dataset does not have severity information. First, four projects, JM1, KC1, KC3, and PC4, are chosen, except for five projects with no severity information, three with zero severity values, and one with too many missing data. Each project consists of several defect related files, the two most important of which are product_module_metrics and defect_product_relations. The former has module_id, error_count, and several metric values for the module. The latter has the information about defects in each module and the severity of those defects. We found that some projects have serious ambiguity problems between these files since the defect information derived from different file combinations gives different results [12]. Therefore, we cannot use the projects with these problems to build prediction models. Finally, we use JM1 and PC4 to build prediction models in this study, since they are the only projects with the least ambiguity problems according to the analysis results in [12].

JM1 is a real time C project which has 315 KSLOC and 10,878 modules. It has 2,102 defective modules and 3,161 defects with *dsl* values from 1 to 5. PC4 is a satellite flight system project implemented in C language, and has 36 KLOC and 1,458 modules. It has 178 defective modules and 370 defects with *dsl* values from 3 to 5.

Most module related metrics in the product_module_metrics file are used as the input metrics for the prediction model. These are divided into three categories: Halstead metrics, McCabe metrics, and LOC-related metrics. The derived metrics obtained from other metrics are not excluded in this study. As shown in Figure 1, the output of the model is the severity-based fault-proneness of a module quantified as input metric values, and this output value is represented in three classes, HSF, LSF, and NF.

Figure 2 and Figure 3 show the distribution of the three module severity metric values of the defective modules in JM1 and PC4, respectively, in the module order based on the MS value. The left vertical axis represents MS and MSD values, and the right vertical axis represents MSL values. For most modules, the MSL value increases as the MS value increases, but the MSD value is not necessarily the same. Especially, the MSD value of PC4 seems to have little relation to the MS value.

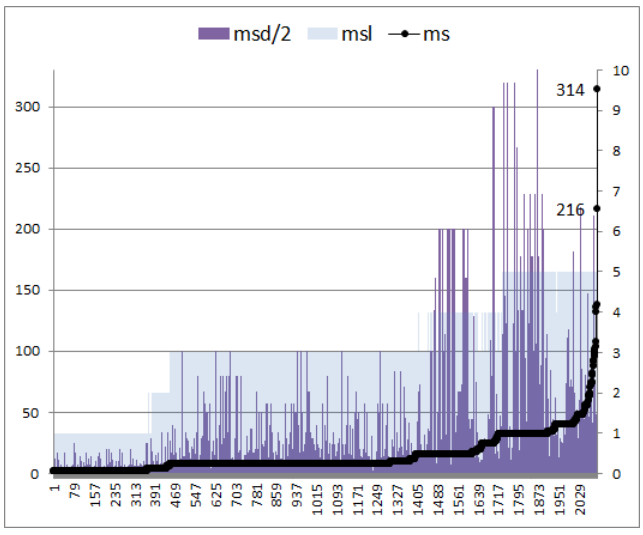

Figure 2. Module Severity based on MS for JM1

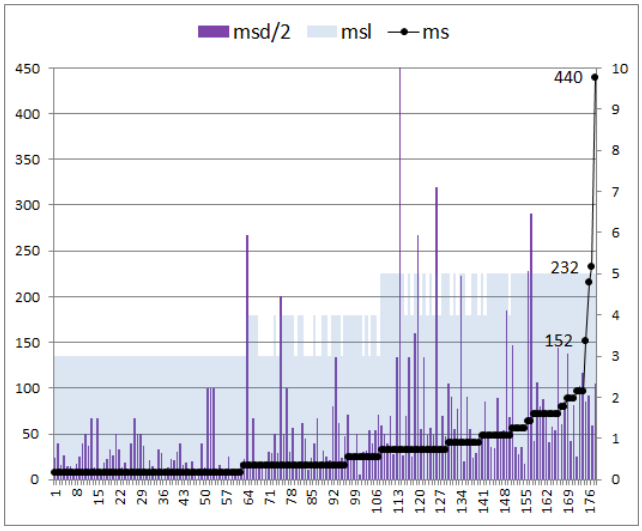

Figure 3. Module Severity based on MS for PC4

Defining output values of a prediction model

As discussed in the previous section, a prediction model has three output class labels in this study. We use three methods that each uses different module severity metric to define severity-based fault-proneness of a module. First, output classes can be defined using MSL values. If MSL_m is 5, then the module m is a HSF module. Otherwise, if MSL_m is in the range of 1 to 4, m is a LSF module. If there are no defects, m is a NF module. That is, HSF module has one or more high severity defects, i.e. defects with *dsl* value of 5, and LSF module doesn't have high severity defects but low severity defects. All previous severity studies used MSL based methods to classify defects. Some of them categorized defects as high severity and low severity [6, 10], but some other studies added medium severity to them [7, 9].

The other two methods for defining the output classes are to use the *MS* and *MSD* metrics, respectively. Since both metrics represent the module severity, the higher the metric value of the module, the more severe the module, and the lower the metric value, the less severe the module. Therefore, the modules are sorted by each metric value and clustered into three groups, HSF, LSF, and NF, based on the metric values. We use simple K-means clustering. Table 1 shows the number of modules belonging to each class as a result of the three different metric based methods. The result distribution of MS or MSD based method seems to be a little unstable compared to the result of MSL based method. Especially, the number of HSF modules of MS-PC4 and MSD-based method is very small. The number of NF modules of the MSL-based method means the number of defect-free modules of JM1 and PC4, respectively. The reason for the higher number of NF modules of MS- and MSD-based methods is that even faulty modules can be considered as NF modules if they have very few defects with very low severity levels.

Table 1. Result distribution of the output values

Method	Project	NF	LSF	HSF
MSL	JM1	8776 (80.7%)	1759 (16.2%)	343 (3.2%)
	PC4	1280 (87.8%)	120 (8.2%)	58 (4.0%)
MS	JM1	9213 (84.7%)	1268 (11.7%)	397 (3.6%)
	PC4	1387 (95.1%)	68 (4.7%)	3 (0.2%)
MSD	JM1	10492 (96.5%)	327 (3.0%)	59 (0.5%)
	PC4	1359 (93.2%)	90 (6.2%)	9 (0.6%)

EMPIRICAL STUDY

Experimental Setting

The dimension of the input data is too high, 21 for JM1 and 36 for PC4, and data in the same metric category can correlate each other. Therefore, we use correlation based feature selection (CFS) to remove the irrelevant attributes. CFS evaluates a subset of attributes by considering the predictive ability of each attribute along with the degree of redundancy between them [13]. Table 2 shows the selected attributes for each model. We use the Experimenter of WEKA [14], a data mining tool widely used in the prediction model studies, for model building and performance evaluation, and use both cases of metrics with and without attribute selection as input metrics. In order to construct the model, we use three classification algorithms, multi-layer perceptron neural network (MLP), naïve bayesian network (NB), and J48 decision tree, which have been the most used in prediction model studies [3].

Table 2. Results of the attribute selection

Method	Project	Selected attributes
MSL	JM1	LOC_BLANK, LOC_CODE_AND_COMMENT, LOC_COMMENTS, CYCLOMATIC_COMPLEXITY, DESIGN_COMPLEXITY,ESSENTIAL_COMPLEXITY, HALSTEAD_CONTENT, LOC_TOTAL
	PC4	LOC_CODE_AND_COMMENT, PARAMETER_COUNT, PERCENT_COMMENTS, MODIFIED_CONDITION_COUNT
MS	JM1	LOC_CODE_AND_COMMENT, LOC_COMMENTS, LOC_TOTAL, CYCLOMATIC_COMPLEXITY, DESIGN_COMPLEXITY, ESSENTIAL_COMPLEXITY, NUM_OPERATORS, NUM_UNIQUE_OPERANDS, HALSTEAD_CONTENT
	PC4	LOC_CODE_AND_COMMENT, LOC_TOTAL, CYCLOMATIC_COMPLEXITY, DECISION_DENSITY, PERCENT_COMMENTS, HALSTEAD_VOLUME
MSD	JM1	HALSTEAD_DIFFICULTY, LOC_TOTAL
	PC4	LOC_CODE_AND_COMMENT,PARAMETER_COUNT

Prediction performance evaluation of the multi-classification model is more difficult than the binary classification model since the performance measures are mostly used for binary models. Therefore, we select accuracy (ACC) which is the percentage of correct predictions for the total input data, and additionally select area under ROC curve (AUC) which has been evaluated as a very important performance measure in recent studies [3, 15, 16]. ROC curve is a two-dimensional graph in which true positive rate is plotted on the Y axis and false positive rate is plotted on the X axis, and it depicts relative tradeoffs between benefits (true positives) and costs (false positives) [17]. When AUC is calculated, three output classes are converted to two classes, HSF and not HSF, since HSF is the most important class to be predicted correctly.

The initial parameter values of each model are based on the default values of WEKA. 10-fold cross validation was repeated 10 times to increase the reliability of the experiment. That is, one model is executed 100 times since it repeats 10 times of 10 times while changing training data set and test data set. Finally, the evaluation results of each model are the average of 10 cross-validation results.

Performance evaluation

Table 3 and Table 4 show the performance evaluation results for JM1 and PC4, respectively. MS and MSD models are new prediction models using module severity metrics *MS* and *MSD* respectively, and MSL model is a typical existing model proposed in previous studies. JM1-CFS and PC4-CFS means the data set with CFS attribute selection, and the number in parentheses indicates the standard deviation of the measure. The results of the MSL model are very similar to those of our previous study [10].

First, when we use JM1, all MS models perform better than the corresponding MSL models in all comparisons of ACC and AUC. The MSD models show better performance than the MSL and MS models except for the ACC results of the NB-JM1 and the J48-JM1-CFS. Nonetheless, some serious results reduce the reliability of the performance results of the MSD models. Comparing the results of JM1 and JM1-CFS to see if CFS affects the performance of the models, there are no significant differences between the MSL models, and the MS models also have no significant differences except that the AUC of J48 increases from 0.56 to 0.64. However, the MSD models show a tremendous increase in the ACC result of NB and a significant decrease in the AUC result of J48. It is interesting to compare the performance of the three classification algorithms using the same data set. In the MS models, as in the MSL models, MLP performs better than the other two algorithms. However, in the MSD models, MLP and J48 for JM1 and MLP and NB for JM1-CFS have similar performance.

When we use PC4, the MS models perform better than the corresponding MSL models except for the AUC result of J48-PC4-CFS. In the ACC comparisons, the MSD models show better performance than the MSL models except for the serious results of the NB-PC4 as in the NB-JM1 case, and show slightly worse performance than the MS models. In the AUC comparisons, the MSD models show worse performance than the MS models, and show worse results than MSL models in the case of PC4-CFS. The performance changes due to CFS attribute selection are larger when using PC4 than JM1. There are no significant differences between the MSL models. However, in the MS models, ACC and AUC increase in NB and AUC decreases in MLP. In the MSD models, AUC decreases in all three algorithms and ACC of NB increases. Among the classification algorithms, MLP shows the best performance when considering both ACC and AUC.

The MSD models show unstable performance results in some cases compared to the MS models, and this problem is more severe when using PC4 than JM1. An important reason for these results is that the dispersion of the MSD values is much larger than that of the MS values since there are extreme outlier modules with very low LOC values. We found several outlier modules with LOC of 1 in PC4, but with severe defects, and that their small LOC values make the MSD of the module very large. In Figure 3, the largest MSD value is 1066, which is much larger than the second largest value of 640. The average of the MSD values of PC4 is 14.162, but the standard deviation is 58.969 due to these outliers. In addition, JM1 is more suitable as an experimental data set, since the code size and number of modules in JM1 are about 10 times larger than those of PC4. Furthermore, while PC4 has *dsl* values in the range of 3 to 5, JM1 has *dsl* values in the range of 1 to 5, so that defects with various severities are distributed to the modules.

Table 3. Prediction Results for JM1

Model	Data Set	Performance Measure	MLP	NB	J48
MSL	JM1	ACC	80.75(0.51)	78.70(0.87)	78.51(0.93)
		AUC	0.76(0.03)	0.76(0.03)	0.53(0.07)
	JM1-CFS	ACC	80.73(0.36)	78.64(0.92)	79.98(0.83)
		AUC	0.76(0.03)	0.75(0.04)	0.60(0.07)
MS	JM1	ACC	84.71(0.23)	81.58(0.74)	82.79(0.71)
		AUC	0.79(0.04)	0.78(0.04)	0.56(0.06)
	JM1-CFS	ACC	84.71(0.22)	81.70(0.69)	83.96(0.51)
		AUC	0.78(0.04)	0.77(0.04)	0.64(0.06)
MSD	JM1	ACC	96.45(0.04)	19.58(1.27)	96.51(0.14)
		AUC	0.86(0.05)	0.86(0.05)	0.80(0.11)
	JM1-CFS	ACC	96.45(0.04)	96.45(0.04)	96.45(0.04)
		AUC	0.79(0.07)	0.86(0.04)	0.50(0.00)

Table 4. Prediction Results for PC4

Model	Data Set	Performance Measure	MLP	NB	J48
MSL	PC4	ACC	87.97(2.17)	64.80(6.65)	86.85(2.03)
		AUC	0.89(0.05)	0.79(0.06)	0.59(0.15)
	PC4-CFS	ACC	88.05(1.34)	86.21(2.03)	88.07(1.67)
		AUC	0.89(0.05)	0.86(0.06)	0.87(0.04)
MS	PC4	ACC	95.38(1.56)	87.37(2.33)	95.17(1.28)
		AUC	0.97(0.04)	0.83(0.24)	0.78(0.25)
	PC4-CFS	ACC	95.48(1.17)	91.33(2.27)	94.88(1.14)
		AUC	0.91(0.19)	0.98(0.04)	0.75(0.25)
MSD	PC4	ACC	93.12(1.55)	54.00(5.41)	92.36(1.60)
		AUC	0.90(0.12)	0.81(0.19)	0.66(0.25)
	PC4-CFS	ACC	93.04(0.75)	92.81(1.27)	93.17(0.28)
		AUC	0.70(0.22)	0.71(0.28)	0.51(0.05)

Finally, we found that the MS models outperform the existing severity-based prediction models, i.e., the MSL models, which simply define model output values, whether the module has high severity defects or not. The MSD models also show better performance than the MSL models in JM1, but show sensitive performance in PC4. These findings imply that the module severity metrics, such as *MS*, are more strongly related to the input metrics of the prediction model than the previous methods that define the severity-based fault-proneness classes of the module.

CONCLUSION

There has long been a strong interest in software fault prediction since the early identification of trouble spots can greatly reduce system development costs. More recently, several studies have shown that severity-based fault-proneness prediction models are much more useful than binary classification models that simply determine whether a module is fault-prone or not.

We proposed new severity-based prediction models using *MS* and *MSD*, two module severity metrics that we had proposed in our previous study. The proposed models are different from the previous models by defining the output value of the model, i.e. the severity-based fault-proneness of the module, using *MS* and *MSD*.

Through an empirical study using the JM1 and PC4 projects in the NASA data set, we found that the MS models outperform the previous prediction models, and the MSD

models show good performance when using JM1. Furthermore, multi-layer perceptron neural network showed the best performance among the three classification algorithms used for model building. Based on these observations, the module severity metrics are much more suitable for expressing the severity-based fault-proneness of a module than previous methods.

ACKNOWLEDGEMENTS
This work was supported by the Sungshin University Research Grant of 2015.

REFERENCES

[1] IEEE Standard Classification for Software Anomalies, IEEE Std. 1044-2009.

[2] C. Catal, 2011, "Software fault prediction: A literature review and current trends," Expert Systems with Applications, 38(4), pp. 4626-4636.

[3] R. Malhotra, 2015, "A systematic review of machine learning techniques for software fault prediction," Applied Soft. Computing, 27, pp. 504-518.

[4] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, 2012, "A Systematic Literature Review on Fault Prediction Performance in Software Engineering," IEEE Trans. Software Eng., 38(6), pp. 1276-1304.

[5] D. E. Harter, C. F. Kemerer, and S. A. Slaughter, 2012, "Does Software Process Improvement Reduce the Severity of Defects? A Longitudinal Field Study," IEEE Trans. Software Eng., 38(4), pp. 810-827.

[6] Y. Zhou and H. Leung, 2006, "Empirical analysis of object oriented design metrics for predicting high and low severity faults," IEEE Trans. Software Eng., 32(10), pp. 771-789.

[7] Y. Singh, A. Kaur, and R. Malhotra, 2010, "Empirical validation of object-oriented metrics for predicting fault proneness models," Software Quality Journal, 18, pp. 3-35.

[8] S. R. Chidamber and C. F. Kemerer, 1994, "A Metrics Suite for Object Oriented Design," IEEE Trans. Software Eng., 20(6), pp. 476-493.

[9] R. Shatnawi and W. Li, 2008, "The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process," Journal of Systems and Software, 81(11), pp. 1868-1882.

[10] E. Hong, 2015, "Software Quality Prediction based on Defect Severity," Journal of the Korea Society of Computer and Information, 20(5), pp. 73-81.

[11] E. Hong, 2015, "A Metrics Set for Measuring Software Module Severity," Journal of The Korea Society of Computer and Information, 20(1), pp. 197-206.

[12] E. Hong, 2013, "Ambiguity Analysis of Defectiveness in NASA MDP data sets," Journal of the Korea Society of IT Services, 12(2), pp. 361-371.

[13] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, 2016, Data Mining: Practical Machine Learning Tools and Techniques, fourth ed. Morgan Kaufmann.

[14] WEKA (Waikato Environment for Knowledge Analysis) http://www.cs.waikato.ac.nz/~ml/weka/.

[15] T. Menzies, J. Greenwald, and A. Frank, 2007, "Data mining static code attributes to learn defect predictors," IEEE Trans. Software Eng., 33(1), pp. 2-13.

[16] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, 2008, "Benchmarking classification models for software defect prediction: a proposed framework and novel findings," IEEE Trans. Software Eng., 34(4), pp. 485-496.

[17] T. Fawcett, 2006, "An introduction to ROC analysis," Pattern recognition letters, 27(8), pp. 861- 874.