V P KUMAR, et al, International Journal of Computers, Electrical and Advanced Communication Engineering [IJCEACE]™. Thomson Reuters (WOS) Research ID: D-1150-2018, Volume 10, Issue 19, PP: 76 - 82, JAN - JUL' 2021

SOFTWARE BUG PREDICTION USING OBJECT-ORIENTED METRICS

VALLURU PRAVEEN KUMAR 1*, P. VIJAYARAGHAVULU 2

M.Tech Student, 2. Asst.Professor
Dept of CSE,
Sri Annamacharya Institute of Technology and Science, Rajampet, Kadapa.

ABSTRACT

Software quality is the fundamental requirement for a user, academia person, software developing organizations and researchers. In this paper a model for object-oriented Software Bug Prediction System (SBPS) has been developed. This model is capable of predicting the existence of bugs in a class if found, during software validation using metrics. The designed model forecasts the occurrences of bugs in a class when any new system is tested on it. For this experiment some open source similar types of defect datasets (projects) have been collected from Promise Software Engineering Repository. Some of these datasets have been selected for prediction of bugs, of which a few are not involved in model construction. First of all, we have formulated some hypotheses corresponding to each and every metric, and from metrics validation based on hypothesis basis finally 14 best suitable metrics have been selected for model creation. The Logistic Regression Classifier provides good accuracy among all classifiers. The proposed model is trained and tested on each of the validated dataset, including validated Combined Dataset separately too. The performance measure (accuracy) is computed in each case and finally it is found that the model provides overall averaged accuracy of 76.27%.

Keywords. Software bug; metrics; correlation of metrics and bug; software bug prediction

Introduction Software quality is the paramount need for a user, academia person, software developing organizations and researchers. ‘‘Quality is never an accident. It is the result of an intelligent effort’’ [1]. Prediction of software defect can only be possible either on the basis of historical data accumulated during implementation of similar or same software projects or it can be developed using design metrics collected during design phase of software development. Object-oriented (OO) approach is different from the traditional programming approach. It separates data and

control; it is based on objects, each of which is a set of defined data and a set of fixed predefined operations, which are known as methods that can be performed on data. The object-oriented (OO) approach has become a more important cornerstone of software engineering than structural design and functional decomposition. In last three decades the object-oriented technology has been widely accepted and object-oriented development is now so pervasive that there is no longer a question of its viability. OO paradigm provides a new potential and better way to analyse a problem, design a solution and implement it in software engineering. The OO approach provides better reusability, reliability and maintainability than the traditional approach, which is based on functional decomposition. Encapsulation is the attribute of OO System that creates self-contained objects that are very easily incorporated into a new design, which will basically promote reusability [2]. Software metric has been defined by Paul Goodman as ''The continuous application of measurement based techniques to the software development process and all its product[s] to provide meaningful and timely information, together with the use of those procedures to recover that process and its products''. Mostly it is categorized into three broader categories: process metric, product metric and project metric. Process metric is used to improve software development and low product maintainability, e.g., defect removal effectiveness during development, the response time of the fix process and pattern of testing of defect arrival. Product metrics deal with the characteristics of a product like size, complexity, design features, presence of quality level in the product and performance of the product, whereas project metrics cover the number of staff members involved in software development, their staffing pattern throughout the life cycle of the software, cost incurred in development of the project, schedule managed and productivity gained [3].

Background and related work Software quality has been under study for a very long time. Various software quality models are available in the literature. In the last few decades many software developers and researchers studies have been carried out in this area of software quality prediction. Neural networks, fuzzy logic, regression tree, etc. and their applications have been used for software quality prediction.

Chidamber and Kemerer [5] presented a theoretical work focusing on OO metrics at design level of Software Development Life Cycle. These metrics are based upon the measurement theories and are used by well-experienced OO software developers. Chidamber and Kemerer [6] focused on the key requirements of measurement to improve the quality of software with the help of a new metrics suite that consists of six design level metrics named WMC, DIT, NOC, CBO, RFC and LCOM. Emam et al [7] used the Chidamber and Kemerer [5] metrics suite and a subset of Lorenz and Kidd metrics to compute the total impact of class size on validation of OO metrics. The prime consent of the author in this paper was on the prediction of class fault proneness for the faults found in the field. The authors have provided strong evidence showing confounding effect of size in the product metrics to fault-proneness relationship. They have experimentally examined and justified an empirical methodology for examining whether there is a confounding effect. Finally, they have performed a study using a large C?? system to justify the confounding effect that arises. Gursaran and Roy [8] noticed that Weyuker's property 9 is not satisfied by any inheritance metric in the Chidamber and Kemerer [6] metric suite. The authors evaluated inheritance metrics proposed by Brito and Carapuca [9]. The authors showed by building on the metric assumptions and the definitions of concatenation given by Chidamber and Kemerer [6] that a particular class of structural inheritance metrics defined on a diagraph abstraction of the inheritance structure can never satisfy property 9 of Weyuker. Jureczko and Madeyski [10] have presented a review based on process methods, i.e., number of revisions, number of changed lines that are new and number of defects in previous revision. They analysed and said that all of the aforementioned metrics rely on the information corresponding to source code artifact changes. It is also pointed out that some of the changes may not be directly related to the software process used. Finally the authors have shown that process metrics can be an effective addition to software defect prediction modules usually built upon prediction metrics. Jin et al [11] proposed fuzzy c-means clustering (FCM) and radial basis function neural network (RBFNN) to build a prediction model of the fault proneness; RBFNN is used as a classificatory, and FCM as a cluster.

Jureczko and Madeyski [12] have presented an analysis regarding defect prediction using clustering technique on software projects. They have used a data repository with 92 versions of 38 proprietary, academic and open source projects. In this paper Hierarchical and K-mean clustering, as well as Kohonen's neural network, has been used to find the groups of similar projects. Two defect prediction models were created for each of the identified groups. In this study Ckjm tool has been used for the retrieval of all metrics, which will be used for the defect prediction model. JUnit and FitnNess have been used as test tools. The authors have identified two clusters and compared results obtained by other researchers. Finally clustering is suggested and applied.

Methodology

3.1 Data description In this paper, 12 open source similar types of projects (Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-learning, berek, forrest 0.8, Zuzel, Intercafe and Nieruchomosci) have been selected, which are the latest release of their retrospective versions and are collected from Promise Software Engineering Repository, which is available at https://code.google.com/p/promisedata/w/list [15] and shown in table 1. The datasets considered for the experimental work in this thesis are of OO defect projects and Marian Jureczko datasets (https://code.google.com/p/promisedata/wiki/MarianJureczko) [16].

3.2 Data assumption The proposed model will consider and work efficiently only on OO datasets. It is presumed that all the factors that may affect the system in buggy and non-buggy categories will be presented in workable dataset/datasets. All the defect datasets must be related with their different metrics sets and bug values corresponding to each and every instance of the dataset/datasets. If any instance lacks the value of some metrics then in that case all such metrics values will be considered as zero.

3.3 Data validation The different defect datasets collected from Promise repository having duplicate data should be deleted except for one entry. Only then will any module provide unbiased result, and the actual performance of the module may be computed. To avoid duplicate entries, all the datasets mentioned earlier have been validated. The data validation has been performed using the SPSS tool [4]

separately on each dataset taken under study. The same technique has also been applied on Combined_dataset, which is an aggregation of datasets (Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-learning, berek, forrest 0.8, Zuzel, Intercafe and Nieruchomosci) with 3868 instances, and all other datasets used in this thesis separately too to avoid duplicate entries. After data validation applied on the Combined_datset, total unique instances are only 3597 (of validated Combined_- dataset) and remaining 271 duplicate entries have been discarded.

3.4 Metrics description 3.4a Dependent and independent variables: Here, bug is a dependent variable that shows whether there is any bug in a class of the each dataset or not. The independent variables are WMC, DIT, NOC, CBO, RFC and LCOM [6], CA and CE [1], LOC and LCOM3 [17] and NPM, DAM, MOA, MFA, CAM, IC, CBM, AMC, MAX_CC and AVG-CC [12].

Conclusions

The goal of this paper is to investigate and assess the relationship between object-oriented metrics and defect prediction in terms of bug by computing the accuracy of the proposed model on different datasets. It is concluded from all the experiments discussed in the paper that some datasets prompting similar prediction accuracy result and some give different results in terms of accuracy at prediction stage in one dataset. Similar conditions occur when training and testing both are done on all the datasets. This type of variation in the accuracy at prediction (testing) stage may be due to different softwares (datasets) that have been selected in this study, among which all have been developed in different environments, by different staff members in terms of their expertise and skills, by different organizations. Are these organizations ISO certified or not? How frequently the staff members of these organizations switching to different companies? Under what conditions the SRS (Software Requirement Specification) has been prepared? Is the SRS complete, unambiguous, correct, valid, verifiable, traceable, testable, consistent and modifiable or not? And the most important one is design—is the design of the model in which the software (dataset) is developed valid? What about its level? Whether low level and high level design has been done prior to coding stage or not? These are the most important points on which the result will depend. If approximately all these

conditions are fulfilled, then there will a less chance of occurrence of bug, and the quality of the software will improve. Finally we saw that our proposed model provides on average 76.27% accuracy at testing (prediction) stage when training is done on all the datasets.

Future work

If some other real datasets are tested on the proposed model, this will provide the fault-prone classes that are to be tested carefully. The result provided by the model in terms of fault proneness towards a particular class will provide clear-cut guidelines to testers to work on those classes carefully. These guidelines will also save cost in reviewing the whole classes of the new system (software datasets). The conclusions made on the basis of our experimental work shown in section 5 are encouraging from an experimental and practical point of view; this urges further studies to corroborate our findings and conclusions. More investigations are required to sketch any specific pattern.

References

[1] The Rotarian 1991 Rotary International, vol. 159, No.4, ISSN 0035-838X, https://books.google.co.in/books?id= eDIEAAAAMBAJ

[2] Singh Y, Kaur A and Malhotra R 2009 Software fault proneness prediction using support vector machine. In: Proceedings of the World Congress on Engineering, vol. 1

[3] Kan S H 2003 Metrics and models in software engineering, 2nd ed. Pearson Education

[4] Wellman B 1998 Doing it ourselves: the SPSS manual as sociology's most influential recent book. USA: University of Massachusetts Press, pp. 71–78, ISBN 978-1-55849-153-3

[5] Chidamber S R and Kemerer C F 1991 Towards a metric suite for object-oriented design. In: OOPSLA '91 Conference Proceedings on Object-oriented Programming Systems, Languages, and Applications, New York, USA: ACM

[6] Chidamber S R and Kemerer C F 1994 A metrics suite for object oriented design. IEEE Trans. Softw. Eng. 20(6)

[7] Emam K E, Benlarbi S, Goel N and Rai S N 2001 The confounding effect of class size on the validity of objectoriented metrics. IEEE Trans. Softw. Eng. 27(7)

[8] Gursaran and Roy G 2001 On the applicability of Weyuker Property 9 to object-oriented structural inheritance complexity metrics. IEEE Trans. Softw. Eng. 27(4)

[9] Brito A F and Carapuca R 1994 Candidate metrics for objectoriented software within a taxonomy framework. J. Syst. Softw. 26: 87–96 [10] Jureczko M and Madeyski L 2006 A review of process metrics in defect prediction studies. In: Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA, USA, July 17–20