2016 1st International Conference on Innovation and Challenges in Cyber Security (ICICCS 2016)

IMPROVING SOFTWARE QUALITY USING MACHINE LEARNING

Kanika Chandra[1], CSE Department, Dr. A.P.J. AKTU, Uttar Pradesh, India
[1]swarna.saxena@gmail.com

Gagan Kapoor[2], CSE Department, Dr. A.P.J. AKTU, Uttar Pradesh, India
[2]Kapoorgagan03@ymail.com

Rashi Kohli[3], CSE Department, Amity University, Greater Noida, India
[3]rashikohli.amity@gmail.com

Archana Gupta[4] CSE Department, Amity University, Greater Noida, India
[4]archana.archi4u@gmail.com

***Abstract*: Software is an entity that keeps on progressing and endures continuous changes, in order to boost its functionality and maintain its effectiveness.During the development of software, even with advanced planning, well documentation and proper process control, are problems that are countered. These defects influence the quality of software in one way or the other which may result into failure. Therefore, in today's neck to neck competition, it is our requirement to control and minimize these defects in software engineering. Software prediction models are typically used to map the patterns of classes of software that are prone to change. This paper highlights the significant analysis in the area's subject to learn and stimulate the association between the metric specifying the object orientation & the concept of change proneness. This would often lead us to rigorous testing so as to find all kinds of possibilities in the data set.We have two views to be addressed:**

(1) Parameters quantification that affects the quality, functionality and productivity of the software.

(2) Machine learning technologies are used for predicting software

Here, the focus of the research paper is to equate and compare all of learning methods corresponding to performance parameter with its statistical method & methodology which would often results enhanced. Data points are the basis for prediction of models.

Keywords: Software Quality, Change proneness, Receiver operating characteristics (ROC), Empirical validation, Software Defects and Prediction, Software Metric.

I. INTRODUCTION

Software quality means a degree to which software or a process satisfies or associates with the requisites. Software can be summarized to deal with properties such as Complexity which is cyclic in nature, property of cohesion i.e. number of function properties which meet the needs of the customers and thereby provide product satisfaction. Software structures and attributes are categorized in two groups mainly internal and external. The internal quality attributes are those that can be measured during the diverse levels of the development cycle leading to the software SDLC).

There are numerous number of resources such as time, cost & efforts are mostly limited particularly in the development & maintenance phases of software, In efforts to use them effectively and efficiently, there has been extensive researches in the past years to analyze and design the co- relationship between software matrices and its various attributes some of them includes the fault proneness and the feature of maintainability. Maintenance phase of the SDLC is the most important and expensive part. It occupies around 80% of the total cost that is invested in the development of the entire software.

Software metric is measure of the degree to which software inhibits some property. The aim is to obtain objective, quantifiable and reproducible measurements, which have multiple applications in budget planning, cost analysis and estimations, quality assurance testing, software debugging, performance optimization, and optimal task assignments. Another significant terminology which needs to be assessed is Change proneness which exemplifies the probability of a specific entity in software that tends to deflect from its normal properties or could change in future. Predictions related to change prone classes is also helpful in maintaining and testing.

We have chosen two open source software of Android for empirical validation. We have taken two versions of two open source software of Android for analyzing the changes occurring in the both. The changes were mainly encountered in relations of addition, deletion and reformed lines in the classes of the new version. The changes are brought in comparison with the preceding classes results. It has been analyzed that metrics has been formed for all the classes of corresponding software. Data points are generated by combining the change statistics and software metrics.

II. LITERATURE SURVEY

In this research, we study and analyze the performances of systematic regression which are traditionally statistical & many machine learning methods such as Ridor, Random trees, Ordinal class classifier, simple logistics and many more. These methods are used to predict the changes in prone classes. The result is basically obtained by comparing the two approaches. The paper is focused and divided in number of sections. The following section consists of work related with this research, followed by dependent and independent variables, Empirical data collection methodology. And the appropriate results and

978-1-5090-2084-3/16/$31.00 ©2016 IEEE

conclusions are presented in the end of this paper. For this research, we basically used the data obtained from previous version and new version of the same software system. The open source software (Android) is used for the rationale of empirical validation. The series of steps that have been followed are shown in the flow graph.

There are number of studies, research and predications that have been done. The Researcher Han et al. worked basically to envisage the concept of change proneness using an art of UML 2.0 models in order to improvise quality design. He used the concept of behavioral dependency measurement (BDM). BDM acts as a crucial indicator for predicting change proneness. Several others enlisted below are the analysis of the correlations that are associated with the study of relationship b/w defects in software system and couplings. [11]

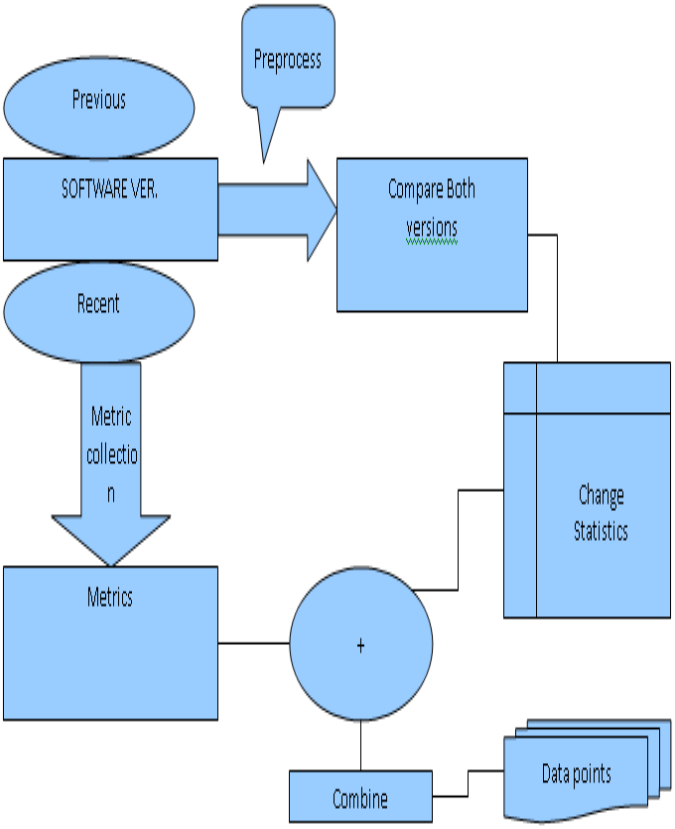

Fig 1.1: Flow Graph Process

III. INDEPENDENT & DEPENDENT VARIABLES

A. Independent variable

In Object Oriented applications, in order to determine, validate and quantify different scenarios of an OO application, OO metrics are implicated throughout SDLC. A single metrics is not adequate to encapsulate all the characteristics required for the process under development. Numerous metrics are required as a whole to evaluate software processes.

Most of the metrics used in our study are the one that holds for various characteristics like size and number of classes, coupling, cohesion, inheritance, encapsulation etc. The concerned metrics value is evaluated using the tool called *Understand for Java.*

B. Dependent variable

Software modification and up gradation are essentials in the development in technology. In order to curb and lower down the cost & labor, the process of prediction and foretelling of change classes can be helpful to achieve the task and thereby it could yield software at lower rates.

In our study & methodology we are meant to study and articulate the correlation between metrics of Object orientation and the concept of change prone classes. In our analyses, the concept of change proneness is featured as the part of dependent variable. To analyze change proneness, receiver operating characteristics (ROC) is considered.

IV. EMPERICAL DATA COLLECTION

Here, we analyze and produce data sets with the help of data collection approach.

We have examined the concept of open source software which is particularly written in the language java referred to as "Android" The terminology of "The open source software" are basically the computer software with code and license that allow to study, develop and change it for some useful purpose. Android is an operating system for mobiles. The description of the software's used is provided in Table 1.

NAME	**VERSION**
Android	4.3.1 , 4.4.2

Table 1.1: Software Description

A. Data collection

In our study, for the purpose of collecting metrics as mentioned before Understand for Java tool have been used. With the help of the software we obtained the metrics which consisted of several OO metrics, size and class metrics. In our study, we are mainly concerned with the metrics for all the classes as we will gauge and predict change in the classes of the same software by analyzing its different versions. Unknown class's metrics are also produced, that can't be accessed and are discarded.

B. Data processing

Data collected from Understand for java tool gave us a result analysis of 225 classes in total with description of any change in comparison to both the version in terms of "yes" or "no".

For the purpose of predicting model we have used WEKA software. Weka is a suite of machine learning software written in java. It is an assortment of visualization tool and algorithm for predicting data and modeling with the combination of graphical user interface.

V. RESULTS

Here the relationship established by the analysis between change proneness of software classes and its metrics are described. Also, we apply some of the machine learning methods to fetch the collective result of software metrics of classes and their change proneness. The data points which are collected from the versions of the software are used for models predictions. Some of the measures which are used for evaluating the performance of each predicted change proneness model are explained below:

For prediction of correct and consistent model, sensitivity and approximation are considered. Sensitivity analysis is the study that shows how the ambiguity in the output of a model can be dispensed and held responsible to different sources of uncertainty in its inputs. In easy term, sensitivity measures the percentage of actual positives which are correctly identified. Specificity or approximation deals with the percentage of negatives which are identified correctly. In ideal terms, to predict change-proneness higher value of sensitivity and specificity is required.**Receiver operating characteristic (ROC)** is a graph plot that depicts binary classifier's performance while its threshold varies. Sensitivity is plotted on the y (vertical)-axis coordinate & specificity on the x (horizontal)-axis coordinate and a curve is obtained from the plotted graph.

While plotting the ROC curves, cut-off points that lie amid the values of 0 and 1 which are selected, and are evaluated on the basis of sensitivity and specificity at each point. The Area under the ROC curve (AUC) is a collectively measure of sensitivity and specificity of the model. AUC value is used compute accuracy. We can find optimal cut-off points by the help of curve of ROC which is an efficient way to ensure quality and performance.

For predicting accuracy, we have applied the model to different sets of data. For this rationale we have carried out the process of ten- cross validation. The data sets are divided in ten subsets which are taken randomly and every time one subset is chosen from ten subsets and the chosen subset are then utilized as the test set or group for further processing. Moreover, the rest of the nine subsets are incorporated to establish training and evaluating set or group. Change proneness of ten subsets can be obtained using this technique.

A. Implementation statistics

The result obtained from the weka software is shown in Table 1.2.

Name of Algorithm	Precision	Recall	ROC Area
Ridor	0.87	0.98	0.61
Random trees	0.96	0.96	0.91
FT	0.99	0.99	0.92
Ordinal class classifier	0.98	0.98	0.91
Simple logistics	0.99	0.99	0.92
Naïve bayes updateable	0.92	0.92	0.93
Voted perception	0.99	0.99	0.63
Part	0.98	0.98	0.92

Table – 1.2 Implementation Statistics for Android

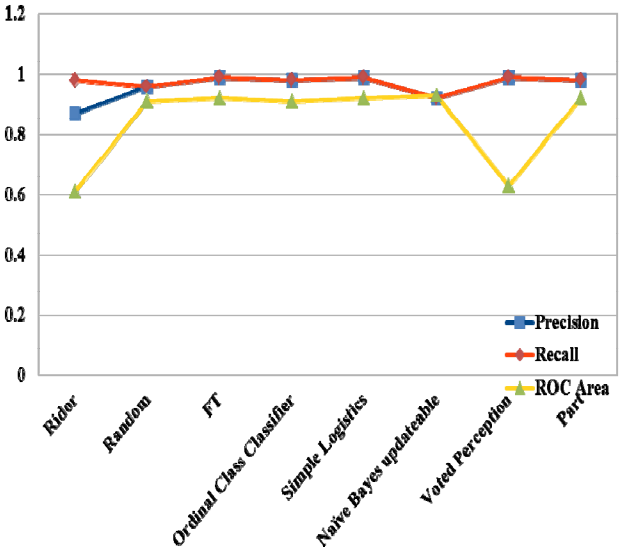

Fig 1.2: Graphical Implementation Result

VI. CONCLUSION

In every software development life cycle, maximum efforts and cost is spent on the testing and maintenance. In our study we have put efforts to calibrate and estimate the change proneness in the classes of the concerned software. With this technique we have proposed a reduction in the efforts that are put in the testing of software. Our goal is to reduce the testing time as with this technique, the developer will have a predetermined notion and idea in the vicinity regarding the sets of classes that poses the changing behaviour and with this knowledge, the developer shall choose these set of classes at the time of testing and will validate the software accordingly.

VII. FUTURE SCOPE

In our work we have compared metrics of classes of different version of object-oriented software & predicted the changes for future versions by comparing the results so that the time invested in testing the software in the software cycle can be reduced to data sets that incur changes and errors and can be performed efficiently and effectively.

Further different techniques can be invented and implemented which would drift the efforts wasted in testing by knowing the area of actual area of faults and errors and performing the testing in a smart, improvised and effective way.

VIII. REFERENCES

[1]. IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology (Vol. 121990). Inst. of Electrical and Electronical Engineers.

[2]. Shepperd, M., & Kadoda, G. (2001). Comparing software prediction techniques using simulation. IEEE Transactions on Software Engineering, 27(11), 1014–1022. http://doi.org/10.1109/32.965341

[3]. Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data Quality: Some Comments on the NASA Software Defect Datasets. IEEE Transactions on Software Engineering, 39(9), 1208–1215. http://doi.org/10.1109/TSE.2013.11

[4]. K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: A replicated case study," Software Process: Improveme & Practice, Vol.16,No.1,2009,pp.39-62.

[5]. A. Porter and R. Selly, "Empirically guided SoftwareDevlopment using Metric-Based Classification Trees," IEEE Software, Vol.7, No.2, 1990, pp.46-54.

[6]. J.R. Quinlan, C4.5:Programs for Machine Learning. Morgan Kaufmann, 1993.

[7].Weka.Available: http://www.cs.waikato.ac.nz/ml/weka/

[8]. Li W, Henry S, Kafura D, Schulman R (1995) Measuring object- oriented design. J Object Oriented Program 8(4):48–55

[9]. Chidamber SR, Kemerer CF (1994) A metrics suite for object oriented design. IEEE Trans SoftwEng 20(6):476–493

[10].Lorenz M, Kidd J (1994) Object-oriented software metrics. In: Prentice Hall object-oriented series. Prentice Hall, Englewood Cliffs

[11] Han AR, Jeon S, Bae D, Hong J (2008) Behavioral dependency measurement for change proneness prediction in UML 2.0 design models, in computer software and applications 32nd annual IEEE international.

[12] L.C. Briand, V.R. Basili, W.M. Thomas, A pattern recognition approach for software engineering data analysis, IEEE Transactions on SoftwareBEngineering 18 (11) (1992) 931–942.