International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Metrics Threshold Analysis On the Basis of Clustering Technique for Fault Prediction

Harlivleen Kaur[1], Prof. Jatinder Singh[2], Prof. Birinder Singh[3]

[1,2,3]Department of Computer Science and Engineering, Baba Banda Singh Bahadur Engineering College
Fatehgarh Sahib,(Punjab) India

Abstract: ***Software metrics encapsulates many software quality factors such as fault proneness, reusability, and maintenance time of the software code. Software metrics are important implements to audit software quality throughout the project lifecycle. Software metrics are numbers accumulated from software code to assess and evaluate where problems are more probable to occur. In this work it is proposed to use a clustering and metrics thresholds based software fault prediction approach and explore it on the dataset, accumulated from Promise Data Repository. Grey relational technique is taken into account as the source for normalized values. The results of the grey relational analysis are then used to conduct fault-proneness classification based on the accuracy (F-measure) of one dataset and compared against the results of another datasets. The result obtain in this work demonstrate the effectiveness of metrics thresholds. This work is validated when the fault labels are unavailable and there is a need to check the accuracy of the software.***

Keywords: Software metrics, CK metrics, Threshold, Software quality

1. Introduction

Software Development is a creative process which is initially difficult to plan and estimate exactly. As most software projects have some risks, deficient information and the required effort. . Software metrics are surrogates of many software quality factors such as fault proneness, reusability, and maintenance effort. Software metrics are numbers collected from software code to assess and evaluate where problems are more probable to happen. These numbers are used to flag warnings of the problematic parts of software code using threshold values. This study focuses on how the combination of clustering and threshold techniques can reduce the potential problems in effectiveness of predictive efficiency due to heterogeneity of the data. This methodology is exploring it on three datasets, collected from Promise data repository. The major focus is on a practical problem that occurs when the fault labels for modules are unavailable. The results of this study demonstrate that the metrics thresholds and clustering is effective than the standalone threshold technique.

2. Literature Survey

Many researchers have carried out consequential work in the area of fault prediction. The literature survey is carried out from the designing of CK metrics to expand different techniques utilized for the modeling of fault prediction. CK metric suit is most commonly used metrics for the object-oriented (OO) software. Chidamber et al. [1] developed and implemented a set of CK metrics for Object Oriented designs. These metrics were mainly focus on quantification theory and reflect the views of trained OO software developers. Basili et al. [2] investigated the suite of object-oriented (OO) design metrics introduced by Chidamber. They define various measures for all metric that represented the expected connection between the metrics and the fault-proneness of the software. Then the hypotheses are tested and prove that some of the metrics were very good predictors, while others were not. Basili also investigated on the results of using the CK metrics suite to predict the quality (fault proneness) of student C++ programs. . Large metric values were found correlated with large number of defects (Cartwright and Shepperd [3]; Subramanyam and Krishnan [9]. In other studies, large values were found correlated with bad design and faults. Erni and Lewerentz [4] have proposed to use the mean and standard deviation of a metric without accounting for the skewness of metric data. Rosenberg has suggested analyzing the effect of the CK metrics on software quality by using histograms and has suggested a set of threshold values for the CK metrics, these values can be used to select classes for inspection or redesign [5]. Shatnawi et al. [6] have used parameters of the logistic regression to identify thresholds for CK metrics based on particular levels of risk.

Nagpal et al.[9] have proposed more efficient estimation sub models. A feature weighted grey relational based clustering method has been integrated with regression techniques. The feature weighted grey relational clustering algorithm uses grey relational analysis for weighting features and also for clustering. The results obtained in their paper showed that clustering could decrease the effect of irrelevant projects on accuracy of estimations. The widely used approach used by the companies is LOC which is old fashioned but easier for assessments [7]. Ammar [10] presents the performance of K-Mean clustering algorithm, depending upon various mean values input methods. The mean values are calculated from the centroid of the particular number of cluster groups. The clustering algorithm consists to two stages with first stage forming the clusters-calculating centroid and the second stage determining the outliers. They investigated the following methods for assigning the mean values in K-Mean clustering algorithm. a) Taking the first k‟ values as centroid. b) Random centroid generation. c) User specified centroid.

R.R. Rathod [11] compares the results obtained with preprocessing by normalization and without preprocessing by normalization of the data set. The basic algorithm detects outer properties in two steps. In first step clusters of original data are formed by utilizing k-mean clustering method. In the second step, it obtain the data elements from each cluster

those are distant from their centers. Then the data elements that are extracted are processed to determine their outlierness with the help of statistical measures. The experiments on dissimilar datasets confirm that preprocessing the data set refines the result. Mamta Mittal, R.K.Sharma and V.P.Singh [12] proposed that Clustering is one of the data mining techniques that divides the database into clusters such that data objects in same clusters are kindred and data objects that associating to different cluster are dissimilar. Analyzers have developed many algorithms for clustering but this paper focus on well defined partitioning technique i.e k-means with threshold based clustering technique. k-means algorithm partition the database into k clusters where k is the parameter defined by user, beside this it is sensitive to outliers and intial seed selection. S. Singh and K.S kahlon [13] derived a threshold metrics value against the bad smell using risk analysis at five different levels. They had taken three versions of Mozilla Firefox as a dataset to validate the study. Their results show that some metrics have threshold values at various risk levels that are of practical use in predicting faulty classes. Finally they designated one threshold value from the various risk levels by determining the largest area under receiver operating character curve for faulty classes at corresponding risk levels.

3. Research Objectives

Research results of earlier studies of mining the faults with better mining techniques are not leading to the acceptable results for fault prediction in software metrics. With the focus on this idea the following objectives of the study are formulated:

- To study existing framework developed for fault prediction.
- To classify the heterogeneous data to homogeneous form with normalization of data set.
- To perform clustering of homogeneous dataset.
- To designate threshold of software metrics for each clustered set of values.
- To analyse the accuracy of designated threshold values by applying it over the other data sets.

4. Proposed Methodology

The methodology used for the prediction of faults in software system is based on the clustering and metrics threshold approach and explore it on datasets collected from Promise data repository. In this work, first step is to normalize the datasets, K-mean clustering techniques namely, centroids (ck) and Euclidean distance for the reduction of heterogeneity. Normalization technique is used to transform all metrics. Clustering methods can be used to group the modules having similar metrics by using similarity measures or dissimilarity measures (distances).

4.1. Data Collection

The PROMISE Repository is a research dataset repository specializing in software engineering research datasets. In this study, experiments are conducted on three versions (1.4, 1.5, and 1.6) of velocity were taken for analysis. These datasets are taken from promise repository which helps us to collect all CK metrics values.

4.2. Tools Used

Weka is a collection of machine learning algorithms for data mining tasks [14]. The algorithms can either be applied directly to a dataset or taken from your own Java code.

SPSS (Software Package used for Statistical Analysis) is a Windows based program that can be used to perform data entry and analysis and to create tables and graphs. SPSS is capable of handling sizably voluminous amounts of data and can perform all of the analyses covered in the text.

4.3. Clustering of Dataset:

In this methodology, to reduce the heterogeneity that exists in the datasets, the initial focus is on clustering of dataset. Clustering is performing with the weka tool. The four main steps involved are:

- Step 1: Normalization: Each data series is normalized according to formula of data processing, so that they have same degree of influence on the dependent variable.
- Step 2: Clustering the datasets based on the normalized data those found using grey relational analysis.
- Step 3: Threshold analysis of the clustered datasets.
- Step 4: After that accuracy is checked by calculating precision and recall.

Grey Relational Grade:

GRA is used to quantify all the influences of various factors and the relationship among data series that is a collection of measurements. The main steps involved in the process are:

Data Processing: The first step is the standardization of the various attributes. Every attribute has the same amount of influence as the data is made dimensionless by using various techniques like upper bound effectiveness, lower bound effectiveness or moderate effectiveness. Upper-bound effectiveness (i.e., larger-the-better) is given by:

$$X(k) = \frac{x_i(k) - min_i x_i(k)}{max_i x_i(k) - min_i x_i(k)} \quad (i)$$

Where i=1,2,….,m and k=1,2,…n.

$$\gamma(x_0(k), x_i(k)) = \frac{\Delta_{min} + \zeta\Delta_{max}}{\Delta_{vj}(k) + \zeta\Delta_{max}} \quad (ii)$$

where;

$\Delta_{0,i}(k) = |x_0(k) - x_i(k)|$ is the difference of the absolute value between $x_0(k)$ *and* $x_i(k)$;
$\Delta_{min} = min_j min_k |x_0(k) - x_j(k)|$ is the smallest value of $\Delta_{0,j} \forall_j \in \{1, 2, \ldots, n\}$;
$\Delta_{max} = \max_j \max_k |x_0(k) - x_j(k)|$ is the largest value of $\Delta_{0,j} \forall_j \in \{1, 2, \ldots, n\}$; and
ζ is the distinguishing coefficient, $\zeta \in (0, 1]$.

The ζ value will change the magnitude of $\gamma(x_0(k), x_i(k))$. In this study the value of ζ has been taken as 0.5.

4.4 Threshold Derivation:

In this work, it is propose to use grey relational values to find threshold using the mean and standard deviation. All metrics data are converted to GRG values and then the parameters, the mean, and the standard deviation are calculated. To find a threshold value for a metric using the distribution parameters, the following calculations is used:

$$T = \mu + \Omega \quad (iii)$$

Where μ is the mean and Ω is the standard deviation

$$\mu = \frac{x_1 + x_2 + \cdots + x_n}{n} \quad (iv)$$

$$\Omega = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\mu - x_i)^2} \quad (v)$$

These threshold values can be used to detect where more faults could be introduced. The values of T are produced for the system under investigation for all six metrics. The derived metrics are then evaluated to identify the faulty classes, and to analyze the accuracy of the designed threshold, it is applied over the other datasets. On the basis of the threshold value, we classify the classes into two groups: faulty classes, if a metric value >= T and not faulty classes, if a metric value <T. Classes in the first group are considered more fault prone than the second group. The classes that exceed a threshold value can be selected for more testing to improve their internal quality.

4.5 Evaluation Parameters:

After the prediction of fault-proneness data, a confusion matrix, which is useful in the process of performance evaluation, has been calculated.The confusion matrix is used to measure the performance of using threshold model in identifying actual fault classes using three measures, Recall, Precision, and F-measure. The actual labels of data items are placed along the rows while the predicted labels are placed along the columns.

Table 4.5.1: Confusion Matrix

	Predicted labels	
Actual labels	False (Non Faulty)	True (Fault)
False (Non Faulty)	True Negative	False Positive
True (Fault)	False negative	True Positive

These measures are calculated as follows:

$$Precision = \frac{TP}{TP+FP} \quad (vi)$$

$$Recall = \frac{TP}{TP+FN} \quad (vii)$$

$$F - measure = \frac{(\beta+1)*Precision*Recall}{\beta*Precision+Recall} \quad (viii)$$

5. Results and Discussions

Threshold values provide a meaningful interpretation for metrics and provide a surrogate to identify classes at risk. The threshold values derived in this study will help to predict the faulty classes. The result shows the practical threshold values only for WMC, DIT, NOC, CBO, RFC, and CBO metrics for predicting the accuracy in the three releases of velocity. Many experimental studies have developed the metrics model or used the metrics values to predict the quality of the software both for design deviance and for error proneness. Study includes the discussion of results in comparison to previous studies done by Rosenberg [26]. In the following, the results are discussed for each metric separately.

- WMC thresholds: WMC is an indicator of a class or interface complexity. Large values of WMC should be considered problematic. There are many threshold values reported previously for WMC, there is still no consensus on a particular value. Table II shows the calculated thresholds values after GRG and actual threshold values on the basis of which performance is evaluated for the velocity 1.4. The mean value is 8.8 and median is 5 and mode is 5.
- DIT thresholds: DIT is an indicator of depth of inheritance in a class. Large DIT values lead to more complex classes, which are difficult to understand, maintain, and reuse. DIT threshold can be used to mark classes that need more attention during both testing and maintenance phases. Although there have been many studies on the impact of the depth of inheritance, there is no consensus on the acceptable depth. Table II shows the potential threshold values of velocity 1.4 for DIT. Table III summarizes the descriptive statistics for DIT threshold values. It is noticed that the mean value for the DIT metric is 2 and close to median and mode.

Table I: Skewness statistics of velocity 1.4.

Metric	Skewness
WMC	-2.41
DIT	-1.41
NOC	-8.28
CBO	-1.28
RFC	-1.34
LCOM	-3.45

WMC, weighted methods complexity; DIT, depth of inheritance hierarchy; NOC, number of child classes; CBO, coupling between objects; RFC, response for class; LCOM, lack of cohesion of methods.

Table II: Threshold values for all metrics of velocity 1.4.

Metric	Threshold value after GRG	Actual Threshold values
WMC	1.006	2
DIT	0.99	2
NOC	1.050	17
CBO	0.934	8
RFC	0.968	15
LCOM	1.069	33

WMC, weighted methods complexity; DIT, depth of inheritance hierarchy; NOC, number of child classes; CBO, coupling between objects; RFC, response for class; LCOM, lack of cohesion of methods.

- NOC thresholds: NOC is an indicator of both abstraction and inheritance in classes. Large values of NOC could be marked as problematic in maintenance and testing phases and could increase the effort required for both phases. NOC thresholds have been considered in many previous works, and corresponding threshold values were identified. Table II shows the potential threshold values of velocity 1.4 for NOC. Table III summarizes the descriptive statistics for NOC threshold values found.
- CBO thresholds: CBO is a well-known metric for measuring coupling among classes. Coupling is considering a problem maker in both the design and code. High coupling makes a class more complex, difficult to understand and increases testing and maintenance efforts. The developers can reduce coupling among classes throughout code cleaning and refactoring continuously. However, where high coupling that should be marked by the developers is not certain. There have been many threshold values reported previously for the CBO. Table II shows the calculated thresholds values after GRG and actual threshold values on the basis of which performance is evaluated for the velocity 1.4
- RFC threshold: RFC is an indicator of the amount of responsibility amounted to a class. Classes that have large response set are prone to faults and need more maintenance. Such classes usually need more testing effort to make sure the class implementation meets all requirements assigned to the classes. The response set should be limited with a threshold values for all classes. RFC values have a large variance, and there is no one threshold found suitable for all projects. Table II shows the actual threshold values after GRG. Table III summarizes the descriptive statistics for RFC threshold values found for the various versions of velocity.

Table III; Descriptive statistics for threshold values in velocity 1.4.

Metric	Minimum	Maximum	Median	Mean	Mode
WMC	0	148	5	8.8	5
DIT	1	4	2	1.8	1
NOC	0	1	0	0.42	0
CBO	0	65	7	10.6	10
RFC	0	236	18	23.6	22
LCOM	0	1188	4	46.1	0

WMC, weighted methods complexity; DIT, depth of inheritance hierarchy; NOC, number of child classes; CBO, coupling between objects; RFC, response for class; LCOM, lack of cohesion of methods.

On the basis of the threshold value, the classes are classified into two groups: faulty classes, if a metric value >= T and not faulty classes, if a metric value <T. Classes in the first group are considered more fault prone than the second group. The classes that exceed a threshold value can be selected for more testing to improve their internal quality. Then the bugs of the metrics are collected from promise repositories of the project. Promise repositories analyze the history of the classes by studying the code repositories. Then compare the faulty or non faulty results with the bugs. From this, confusion matrix is created. The confusion matrix is used to measure the performance of using thresholds model in identifying actual fault classes using three measures, Recall, Precision, and F-measure. According to many researchers the higher value or value close to 1 of F-measure is considered as the best case and the value close to 0 is considered as the worse case. So the higher values of F-measure for each metric is more preferable. Then these values are evaluated according to their True Positive rate and False Negative rate.

After comparing the TP and FN rate, results are concluded. If the count of True Positive is more, then it is concluded that the bugs are truly detected and if the false negative count is more, then it is proved that there are no bugs in those classes and this study also gives the results without bugs. The values of both the Recall and Precision are between 0 and 1. Values that are close to 1 mean better results. If the value is 1, then the classifier is ideal and without FN or FP. The high values of Recall and Precision do not coincide. In practice, it is hard to achieve high Recall and Precision, that is, high Recall occurs often with low Precision. Then the F value of different versions of each metric are compared and the above results are concluded. The results show that the performance of most of the systems is evaluated by their derived thresholds and then apply it over the other datasets. Following table show the values of precision, recall and their corresponding F-measure.

Table IV: Values of Precision, Recall and F-measure of velocity 1.4.

Metric	Precision	Recall	F-measure
WMC	0.66	0.92	0.76
DIT	0.63	0.44	0.51
NOC	1	0.007	0.013
CBO	0.78	0.59	0.67
RFC	0.84	0.65	0.73
LCOM	0.19	0.72	0.29

WMC, weighted methods complexity; DIT, depth of inheritance hierarchy; NOC, number of child classes; CBO, coupling between objects; RFC, response for class; LCOM, lack of cohesion of methods.

Software quality is a vital part of the software engineering, and tools are important to audit and assess the software quality. Software quality is assessed indirectly by measuring the internal structure of software using validated metrics. There are many software metrics that were proposed to measure the internal structure of object-oriented systems. This work is limited to only one set of software metrics, the CK metrics, which were proposed and validated as measures of six internal attributes of the software classes including coupling, cohesion, complexity, inheritance depth, inheritance breadth, and class responsibilities. Many commercial and open-source tools are available to assess and provide quality indicators of software code using software metrics. Most of these tools collect metrics and analyze metrics using software threshold values that are already injected in these tools. Software metrics can be used to evaluate software quality

from various perspectives including software fault proneness and maintenance effort.

The results obtained in previous work do not report the same threshold for a metric. It is noticed that previously reported thresholds are not similar and different from findings in this work. However, this work is distinguished from the previous work in the following points:

- The proposed techniques in previous works could not identify thresholds for all CK metrics.
- The identified thresholds using data distribution (histogram analysis and distribution parameters) were based on either small number of projects or no empirical work has been conducted to identify thresholds.
- The effect of scenes in metrics has not been accounted for although data skewness has been reported frequently.

Following values of threshold are chosen as the best performance threshold values. The below table describe the values of proposed threshold of every metric:

Table V: Proposed threshold values

Metric	Proposed Threshold values.
WMC	2
DIT	2
NOC	17
CBO	8
RFC	15
LCOM	33

WMC, weighted methods complexity; DIT, depth of inheritance hierarchy; NOC, number of child classes; CBO, coupling between objects; RFC, response for class; LCOM, lack of cohesion of methods.

6. Conclusion and Future scope

Finding where quality can be improved is a vital issue in software quality. Appropriate metric tools are needed to identify the classes that are more fault prone during both development and testing phases. This work resolves the heterogeneity problems that exist in the datasets. In order to confirm the effectiveness of proposed work, three different data sets have been used for fault prediction. The statistical results showed better fault classification using clustering based threshold values. It is suggested to use clustering technique on software metrics before assessing software quality. The results confirm that the proposed feature weighted grey relational clustering algorithm performed appreciably for software effort estimation.

In the future, it is plan to apply hierarchical clustering methods such as agglomerative clustering and fuzzy clustering methods on these datasets. Future work will consider comparing K-means clustering and thresholds based approach to different clustering based approaches.

References

[1] Shyam R. Chidamber and Chris F. Kemerer (1994), "A Metrics Suite for Object Oriented Design," IEEE transactions on software engineering, Vol. 20, No. 6.
[2] V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object- Oriented Design Metrics as Quality Indicators," IEEE Trans.Software Eng., vol. 22, no. 10, Oct. 1996.
[3] Cartwright M, Shepperd M (2000), "An empirical investigation of an object-oriented software system". IEEE Transactions on Software Engineering.
[4] Erni K, Lewerentz C. (1996), "Applying design–metrics to object–oriented frameworks". Proc. of the Third International Software Metrics Symposium; pp. 25–26.
[5] Rosenberg L. (1996), "Metrics for object oriented environment." Proc., EFAITP/AIE 3rd Annual Software Metrics Conference.
[6] Shatnawi R. (2010), "Quantitative investigation of the acceptable risk levels of object–oriented metrics in open–source systems". IEEE Transactions on Software Engineering, Vol. 27, No. 4, pp. 216–225.
[7] Duda, R.O., & Hart, P.E. (1973), "Pattern classification and scene analysis", John Wiley &Sons, Inc.,New York.
[8] Subramanyam R, Krishnan M. (2003) "Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects". IEEE Transactions on Software Engineering, pp. 142-150.
[9] Geeta Nagpal, Moin uddin and Arvinder kaur (2013), "Estimating Project Development Effort Using Clustered Regression Approach". CCSIT, SIPP, AISC, PDCTA - 2013 pp. 493–507.
[10] Ammar. W. Mohemmed (2010)," Particle Swarm Optimization for Outlier Detection", Vol. 07, pp. 220-231.
[11] R. R. Rathod and Dr. B. F. Momin (2013)," Performance evaluation of Outlier Detection with Normalized Data Set", Department of Information Technology Walchand College of Enginee- ing Sangli, Maharashtra State, India, pp. 134-144.
[12] Mamta Mittal, R.K.Sharma and V.P.Singh(2014)," Validation of *k*-means and Threshold based Clustering Method" International Journal of Advancements in Technology. Vol. 5, No. 2, pp. 153-160.
[13] Satwinder Singh , K.S Kahlon (2014) , " Object oriented software metrics threshold values at quantitative acceptable risk level " Springer, Vol. 2, No. 3, pp. 191–205.
[14] http://www.cs.waikato.ac.nz/ml/weka/