http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 13 (8): 1540-1547, 2014
ISSN 1812-5638 / DOI: 10.3923/itj.2014.1540.1547

Predicting Maintainability of Object-oriented Software Using Metric Threshold

[1,2]A.D. Bakar, [1]A. Sultan, [1]H. Zulzalil and [1]J. Din
[1]Faculty of Computer Science, University Putra Malaysia, Serdang, 43400, Selangor, Malaysia
[2]Department of Computer Science, State University of Zanzibar, P.O. Box 146, Zanzibar, Tanzania

Abstract: This study presents the empirical investigation into maintainability of software using Chidamber and Kemmerer Metric suite. The study used metric threshold to propose the model for predicting maintainability of object-oriented software. Two geospatial software systems were used to identify the extent on which the metrics in Chidamber and Kemmerer suite can be used to track the needed efforts during maintenance phase. Linear Discriminant Analysis was used to find the performance measurement for each metric in order to understand their overall effect on particular software product. The results indicated the significant impact of the Chidamber and Kemmerer metrics in predicting maintainability when threshold are used over experts' opinions. Moreover, the results found the important role of size and inheritance metrics in predicting maintainability of Object-oriented software and highlighted the needs for further empirical investigation. This is specifically on the production of more metrics thresholds that give researchers and practitioners a room to work on more metrics.

Key words: Maintainability, metric threshold, expert's opinions, object-oriented software, linear discriminant analysis

INTRODUCTION

Software engineering is associated with the principles and procedures of software development in terms of processes, resources and products. Besides, Summerville (2011) listed maintenance as one of the stages in the software development life cycle and the critical component that consumes substantial time and resources (Pigoski, 2006; Soi, 1985). For that reason, many researchers pay more attention when dealing with maintainability as it is the quality issue in both managerial and technical aspects (Vivanco and Pizzi, 2004; Dagpinar and Jahnke, 2003; Mizuno and Hata, 2010; Ma *et al*., 2013). In order to have better understanding about the extent of the efforts during the maintenance phase, researchers and other practitioners use software metrics to quantify the software quality attributes (Preece, 2001; Muketha *et al*., 2010). Using either single software metric or group of software metrics called "suite", researchers and other practitioners provide a way to understand various quality attributes and their effect in software products.

However, several quality models have been proposed as a strategies for predicting different software quality attributes (Emam, 2002; Pai and Dugan, 2007; Zhang, 2013). Yet, those models rely on human opinions for collecting the input data during development or validation stage. This subjective process criticized for the production of inaccurate results. Bias is among the leading critics whereby, the results from the experts are always depending on what is on the respondents' mind at a particular time. The respondent can give different answers for the same questions in different time at different situation (Knauf *et al*., 2007).

The study presented a preliminary stage of the research to develop the quality model in predicting the maintainability of object-oriented software. The major research intends to use complete Chidamber and Kemmerer (CK) metrics suite as one of the model in the validation process of the proposed model. The researchers were convinced to contribute their findings into the body of software quality perspective.

This study used thresholds for the six software metrics proposed by Chidamber and Kemmerer (1994) as a cut-off-points in classifying defect and non-defect classes. The proposed model was applied to two geospatial software systems and the Linear Discriminant Analysis (LDA) was also used to calculate the classification performance for the metrics. The results show the selection performance measurement of 82% when the metrics subset applied to Geotool and 80% for

Correspomding Author: A.D. Bakar, Faculty of Computer Science, University Putra Malaysia, Serdang, 43400, Selangor, Malaysia

Geoserver. Eventually, it was found that the development of the data-driven models facilitate the classification performance and accuracy.

PROBLEMS WITH EXPERTS' OPINIONS IN DEVELOPING QUALITY MODEL

The experts in different fields have been used as the source of scientific attempts in data interpretation, the prediction of system behaviour, ranking the performance of the models or process and access of uncertainties (Kulkarni, 1985; Leal *et al*., 2007; Krueger *et al*., 2012). In software engineering, this practice is taking place and discussed by several researchers, whereby the practitioners used experts' opinions to develop and to validate their models. However, the practice seemed to provide the noteworthy results, this method is criticised and distrusted as the genuine way to produce accurate model. Adomavicius and Tuzhilin (2001) acknowledge the problem of relaying on experts rather than reading the values from the data because it can lead to wrong results. For that reason, they proposed a mixed approach of experts and data-driven framework to build behavioural profiles of users in solving human weakness problems. Another study on ranking the software engineering metrics by Li and Smidts (2003) also used experts' opinions to propose a framework for selecting the best reliability metrics. The researchers admitted difficulties in identifying software metrics experts due to lack of its maturity in software engineering. Thus, the use of experts ranking opinions is debatable due to its nature of subjectivity.

Knauf *et al*. (2007) observed the inconsistencies of the experts' opinions over time whereby, in a different time, one can receive different results from the similar person in a particular area. Therefore, to solve the problem of bias and human errors, they advised practitioners to reduce the human involvement in software development models practice. Vivanco and Pizzi (2004) developed a Genetic Algorithm (GA) model to predict maintainability of software using experts in representation phase. They admitted that the experts' dependent ranking was one of the threats to the validity of their model due to the subjectivity of opinions. To elaborate more about the subjectivity of data acquisition, the paragraph below analyzes the steps of data eliciting from experts by Boring *et al*. (2005) and Knol *et al*. (2010). The purpose is to show how the proposed steps of getting information from an expert can also lead to wrong results. The steps include selection of the experts, training of the experts, elicitation and aggregation of the opinions and the use of the results in making the decisions.

Normally, experts are expected to make judgments based on experiences about a phenomenon. Since the selection of an appropriate expert is also a judgment, most researchers face the challenge on how to understanding the suitable experts for particular field. This is because most of the researchers are too junior in that body of knowledge. Similarly, the new emergent field like evolutionary computation are faced with the problem of inadequate experts to respond to surveys. After data being collected, they aggregated to be used in making decisions. Since the input was entrusted, the decision and the outcome will also be entrusted. Likewise, the developed model basing on entrusted input can result in wrong purpose, quality focus and not performing intended work.

AN ANALYSIS OF CHIDAMBER AND KEMERER METRICS THRESHOLD

The CK metric suite comprises of six well known individual metrics. The metrics include Weight Method per Class (WMC), Depth of Inheritance Tree (DIT), Number of Children (NOC), Coupling between Object (CBO), Response for Class (RFC) and Lack of Cohesion Methods (LCOM). The acceptance of this metrics suite attracts many researchers. Some of prominent researchers such as Olague *et al*. (2007), Dubey and Rana (2011) and Singh *et al*. (2011a) have used this subset in validating different studies in software engineering discipline. This paper used the CK metrics suite for three major reasons:

- Software metrics in the CK suite have been tested and validated more than other metrics in software engineering paradigm from the earliest time of their invention up to the present (Basili *et al*., 1996; Antony, 2013)
- The software metrics in the CK suite are almost represents all important maintainability quality attributes. For instance DIT measures the inheritance, CBO measures coupled objects, WMC measures the size of class and LCOM measures the interconnection between methods of class. Consequently, all metrics determine the reusability which is an important feature for facilitating quality in object-oriented systems
- All six metrics in CK suite have their published threshold in the literatures. Their thresholds have been validated through theories, experiments and statistics. Table 1 shows the available threshold for the six CK metrics with their respective references

Table 1: Proposed metric threshold for the CK suite

Metric	Proposed threshold	References
CBO	14	Bouktif *et al.* (2002)
WMC	20	Rosenberg *et al.* (1999), Bernarbi *et al.* (2000) Misra and Bhavsar (2003)
LCOM	20	Ferreira *et al.* (2012)
DIT	2	Ferreira *et al.* (2012)
RFC	44	Shatnawi *et al.* (2010)
NOC	2	Chhikara *et al.* (2011) Herbold *et al.* (2011) Mago and Kaur (2012), Chandra and Linda (2010)

The following subsections provide the overview of each metrics in CK suite with their respective threshold.

Weight method per class (WMC): WMC is a method count inside the object-oriented class. Mathematically, it is the number of methods implemented in a class. The number of methods and the complexity of individual methods determine the time and needed efforts to maintain the software. The larger the number of methods in the object means more methods are likely to have an impact on the derived methods, since the children will inherit all the methods from the main object. The larger number of methods may result in the formulation of the application specific objects. For that reason it limits the possibility of reuse. In this case, the product with more methods is difficult to maintain since one method can call information from more than one method inside and outside the class. Studies also suggest software product with more number of methods for particular class leads to increase in bugs and the time to fix them (Singh *et al.*, 2011b; Misra and Bhavsar, 2003). Thus, maintainability decline with the increase of WMC. Practitioners recommended the maximum number of methods in one class should not exceed 20 (Rosenberg *et al.*, 1999; Bernarbi *et al.*, 2000; Misra and Bhavsar, 2003).

Depth of inheritance tree (DIT): This is the maximum birthright pathway from the class to the base class in the inheritance composition of the system. The base class value of the DIT is equal to zero. This means that the number of the class definition lengthens the basis of its hierarchy. The deeper the hierarchy of the class, the more methods and variable are likely to be inherited and also the more the complexity of the product since it is very difficult to understand where the method and its defined fields. Although, inheritance implemented to increases the reuse philosophy inside the system but the higher the tree could increase the probability of the faults and leads to decrease in maintainability index. Moreover, the maintainers need more time and efforts to trace the sequence of the class up to the end of the tree in order to understand what the class is doing. According to Ferreira *et al.* (2012), the recommended DIT in single class is less than 2.

Number of children (NOC): Like DIT, NOC is one of the classifications of level-oriented design in object-oriented software development. It is a number of immediate subclasses (children) subordinate to a class (parent) in the class hierarchy. The NOC metric measures how many classes inherit direct methods of the parent class which show the improper abstraction of the parent classes. Increase the number of class tree also increase the class size which may leads to the increases of the class complexity (Chhikara *et al.*, 2011). Therefore, a class with a high NOC indicates complexity at the top of the class hierarchy. The class is potentially influencing a large number of descendant classes. This can be a sign of complexity index that need more efforts in maintaining the software. Herbold *et al.* (2011) found no restriction for these metrics. In emphasis this concept Mago and Kaur (2012) suggested the 0-6 as the interval for the threshold values for NOC. Chandra and Linda (2010) suggested using the equal amount of threshold for both DIT and NOC. Thus, this paper adopted the benchmark value of 2.

Response for a class (RFC): RFC refers to a set of methods executed in response to the message received by an object of that class. Basically, it is the number of sets of methods in a class. RFC is also associated with communication between classes since their responses to the methods are called from outside of the class. Class with higher RFC is more complex to understand since one method can call more than one method from different classes. Therefore, the large number of RFC measure means more decline of maintainability of that class and software product in general. Shatnawi *et al.* (2010) recommended the threshold for RFC to be 44.

Coupling between object class (CBO): CBO is closely related to the RFC. It is the degree of dependency between the object within a class. A class is supposed to be coupled to another class if it accesses one or more of its variables or invokes at least one of its methods. High coupling is undesirable since extreme coupling between object classes is disadvantageous to modular design. Thus, to improve modularity and encapsulation within the system, inter-object class couples should be kept to a minimum. The more independent a class is, the easier it is to reuse it in another application.

The CBO is a measure of the number of other classes to which methods are coupled. The larger the number of couple objects, the higher the sensitivity to make changes since they depend on each other and increase the maintenance efforts. Thus, the maintainability of the software product decreases with the increases of the CBO. Bouktif *et al.* (2002) and Shatnawi *et al.* (2010) set maximum number of the coupled objects not to exceed 13.

Lack of cohesion methods (LCOM): The cohesion is a measure for the number of unconnected method pairs in a class representing interrelatedness between methods in a class. For the quality and less complicated class, the value of cohesive methods should be more than that of the CBO. The benchmark regarded this metric has been considered based on its relationship with the CBO measure. According to their relationship, the two measures are inversely proportional to each other. Since the coupled objects are recommended to be less than the cohesive objects, then, if the number of coupled objects is n, then the cohesion between object should be greater than or equal to n. (The assumption is that the COM should be greater than CBO). The expression below shows the relationship of the COM and CBO:

$$\text{Coupling} \propto \frac{1}{\text{Cohesion}}$$

LCOM represents the measure of the parts having no cohesion. Maintainability increases with the decrease of LCOM. Thus, for the LCOM and CBO relationship, they are directly related to each other, but LCOM should be equal or less than CBO:

$$\text{CBO} \propto \text{LCOM}$$

Many researchers skipped this metrics in their experiment due to its insignificant results. For example Shatnawi *et al.* (2010) did not identify reliable value in two different software applications. However, Ferreira *et al.* (2012) achieve some significant results on this metrics. They recommended the LCOM threshold to be less than 20 (Assumption is the CBO must not exceed LCOM).

MATERIALS AND METHODS

The study used LDA to find the linear combination of features which separates two or more classes of objects. Software metrics selection is characterized as the classification problem whereby the needed software metrics to measure the quality of software is assumed to be more than one amidst available dozens software. The question is which software metrics are capable for particular software. The precision and recall categorization used to find the performance of the Chidamber and Kemmerer (CK) metrics. Calculation of precision and recall were based on generated information from the metric values that was then encoded to binary digits that presents the complex classes (difficult to maintain) and less complicated ones (easy to maintain) as shown in Fig. 1. The two classifications were regarded as the

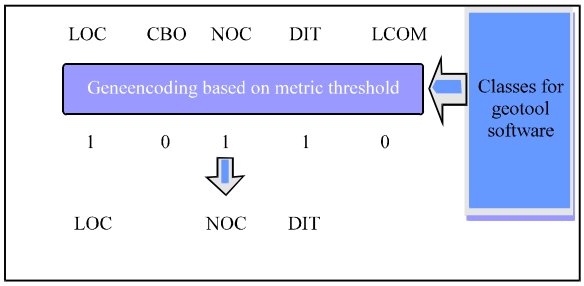

Fig. 1: Encoding of the metrics using thresholds

predictor of fault prone classes and non-fault classes. Researchers have been using these two types of opposite complexity factors as feasible classification in calculating performance accuracy based on the fault prediction (Zhou *et al.*, 2010; Mizuno and Hata, 2010; Sharma *et al.*, 2012).

The software metrics values from the software classes' generated using open-source tool called CK Java Metric tool (CKJM) (Spinellis, 2005). The tool was proved to outperform other metrics values generating tools like the Jdepend, Classycle, CCCC, RSM and ES2 (Lincke *et al.*, 2008; Poornima, 2011). The metric values were generated from the classes of the two geospatial Java software systems. The Geotool contains 2312 classes and the Geoserver has 5530 classes. Empirically, the software were used to investigate the software metrics classification performance to predict maintainability of software based on the defect classes in comparison with the threshold set for different metrics. The two software systems are open-source that enabled the experiment to interact directly with the source code. Thus, the openness of the open-source software (OSS) catalysed the motives behind the use of these ready-made products. Moreover, most of the OSS projects have the opportunities to be tested by many programmers for many times. Thus, they are free from significant bugs.

Experimental analysis: The generated values from six CK metrics were arranged based on acceptable and unacceptable ranges of the metric threshold. Figure 2 shows the results from the exercise of generating metrics values from the software product classes. The task introduced the secondary nominal values 1s, for the acceptable values and 0s for the unacceptable value. The binary values were then used as input for the developed model.

The F-measure evaluation started with the combination of the results categorization. Table 2 below shows the categories of the 2 by 2 contingency table. The expressions below indicate manipulation of two selection

Class_Name	WMC	WMCb	DIT	DITb	NOC	NOCb	CBO	CBOb	RFC	RFCb	LCOM	LCOMb
TranscoderOutput	16	1	1	1	0	1	13	1	17	1	90	1
TranscodingHints	8	1	3	1	0	1	14	1	26	1	28	1
SVGAbstractTranscoder$SVGAbstract	17	1	0	1	0	1	12	1	48	0	0	0
JPEGTranscoder	4	1	0	1	0	1	14	1	25	1	2	0
PaintKey	2	1	0	1	0	1	2	1	3	1	1	0
ImageTranscoder	7	1	0	1	3	1	15	0	38	1	19	0
ToSVGAbstractTranscoder	5	1	0	1	1	1	13	1	29	1	8	0
StringKey	2	1	0	1	0	1	6	1	3	1	1	0
GdiObject	7	1	1	1	0	1	4	1	8	1	0	0
MetaRecord	6	1	1	1	2	1	5	1	12	1	0	0
svg2svg.SVGTranscoder$NewlineVal	2	1	1	1	0	1	2	1	3	1	0	0
Messages	6	1	1	1	0	1	3	1	15	1	3	0
RecordStore	21	0	1	1	0	1	3	1	36	1	14	0
DOMImplementationKey	2	1	0	1	0	1	2	1	3	1	1	0
WMFRecordStore	11	1	0	1	0	1	7	1	25	1	21	0
PrintTranscoder	14	1	0	1	0	1	14	1	80	0	65	0
Transcoder	8	1	1	1	0	1	8	1	8	1	28	1
OutputManager$NameInfo	1	1	1	1	0	1	2	1	2	1	0	0
JPEGTranscoder$1	0	1	1	1	0	1	2	1	0	1	0	0
TIFFTranscoder$WriteAdapter	1	1	1	1	0	1	3	1	1	1	0	0
WMFFont	2	1	1	1	0	1	4	1	3	1	0	0
OutputManager	27	0	1	1	0	1	4	1	56	0	129	1

Fig. 2: Encoded classes based on metrics threshold

Table 2: Classification accuracy for the metrics selection

	Predicted negative	Predicted positive
Selected (positive)	fp	tp
Not selected (negative)	tn	fn

probability for the correct and incorrect information retrieved, from the metrics category which based on product classes based on software metrics threshold.

The classification accuracy is between 0 and 1 with the expression below:

$$F-measure = \frac{2 \times Recall \times Presision}{Recall + Presision}$$

Where:

$$Recall = \frac{tp}{tp + fn}$$

And:

$$Precision = \frac{tp}{fp + tp}$$

The contents are the values of the maintainable classes associated with the selected metrics and the complexity classes related to the selected metrics. Precision and recall for the six CK software metrics are the probabilities of less complicated classes and complicated classes that were then used to calculate the f - measure. Table 3 shows the performance measurement for the CK metrics in both two applied software products.

Table 3: CK metrics performance when applying to Geoserver and Geotool

	Performance-measurements (f-measures)	
Metrics	Geotool	Geoserver
WMC	0.922244	0.922296
DIT	0.942285	0.917219
NOC	0.977555	0.976600
CBO	0.860521	0.881236
RFC	0.869339	0.866004
LCOM	0.381964	0.223179
CK Suite	0.825651	0.797776

RESULTS AND DISCUSSION

This research aimed to develop the metrics selection model for predicting the maintainability of object-oriented software using metrics threshold. The researchers present the first stage of the ongoing research to develop the metrics selection model in predicting maintainability of object-oriented systems. The CK metric suite was used at

the initial stage as the comparison subject in the validation process of the proposed quality model developed using Genetic Algorithms (GA).

The earlier stage results motivated the researchers to present their findings. The stage was to check the performance of CK metrics suite in predicting maintainability using threshold in comparison with the use of experts' opinion used by Vivanco and Pizzi (2004). The results from both two treatments showed the significant performance of 0.82 and 0.80 for Geotool and Geoserver respectively. The overall results depicted three useful impacts:

- First, results showed the improvement of metrics classification performance using software metrics threshold in predicting maintainability of software based on a complete CK suite
- Second, the results may be used as a comparison base, for the model development whereby, the software metrics obtained from that proposed model were compared
- Third finding is regarding the individual results obtained from the two systems. An assumption is software product design architecture differs from one product to another. The same applied to the metrics required in predicting the quality attributes of those software. Although, the first three selected metrics were the same, but their positions and their performance measurements were different. The selection of similar metrics was because the software products measured were from the same context. We believe that if the model applied to the software products from different context the model will select different metrics for different software products

The researchers did not aim to compare the individual metrics but the results added extra helpful findings. The WMC, DIT and NOC emerged as the leading effective metrics in predicting the maintainability for both software products, although their positions of the metrics differ. In general, the size and inheritance metrics were appeared at the top of the selected metrics list. Wang *et al.* (2011) reported that three metrics are enough to predict the quality of software. In this case the results have added extra finding that the CK suite alone can also be used as the metric selection model in predicting maintainability when objective technique is used. The model can also be used for the selection of the best software system which can easily be maintained. This model can be helpful to software practitioners, especially open source software (OSS) adaptors in making wiser decisions in the situation where several software are available for a particular solution. This is normal scenario in the OSS industry where there are dozens of products for a particular use.

The selection of the inheritance and size metrics in both two subjects conforms to the reported result by Harrison *et al.* (2000) about the difficulties of maintaining the software with inheritance features inside. In a similar context, Singh *et al.* (2011a) found the great impact of size metrics in software quality prediction. The outlying of the LCOM among the top three metrics in our study agrees with the finding reported by Dagpinar and Jahnke (2003) when they observed the cohesion metrics of having limited effect in predicting the quality of software while size possessed the dramatic effect.

CONCLUSION

In this study, the authors presented the preliminary findings of their work which used the CK software metrics suite in predicting maintainability of object-oriented software. The model was developed based on the software metric threshold whereby, the metrics values were categorized in conjunction with the classes of the two geospatial software. The experimental results show the significant performance of the CK metrics suite in the prediction of the maintainability of software when software metrics thresholds are used. Moreover, the experiment identified remarkable finding that there is a possibility to predict maintainability based on the most effective metrics out of the dozens available metrics. In conclusion, the model can be used to aid software practitioners in identifying suitable quality software out of many available software products for a particular purpose. This is more concerned for the open source software practitioners. In the future, the study suggested the need for the researchers and practitioners to work on the production and validation of more software metrics thresholds to give practitioners room to work with more metrics.

REFERENCES

Adomavicius, G. and A. Tuzhilin, 2001. Using data mining methods to build customer pro?les. Computer, 34: 74-82.

Antony, P.J., 2013. Predicting reliability of software using thresholds of CK metrics. Int. J. Adv. Network. Appl., 4: 1778-1785.

Basili, V.R., L.C. Briand and W.L. Melo, 1996. A validation of Object-oriented design metrics as quality indicators. IEEE Trans. Software Eng., 22: 751-761.

Bernarbi, S., K. Emam, N. Goel and S. Rai, 2000. Threshold for object-oriented measures. Proceedings of the 11th International Symposium on Software Reliability Engineering, October 8-11, 2000, Washington, DC., USA., pp: 24-38.

Boring, R., D. Gertman, J. Joe, J. Marble, W. Galyean, L. Blackwood and H. Blackman, 2005. Simplified expert elicitation guidelines for risk assessment of operating events. Idaho National Laboratory Report, Office of Nuclear Regulatory Research, U.S Nuclear Regulatory Commission, Washington, DC., USA.

Bouktif, S., B. Kegl and H. Sahraoui, 2002. Combining software quality predictive models: An evolutionary approach. Proceedings of the International Conference on Software Maintenance, October 3-6, 2002, Canada, pp: 385-392.

Chandra, E. and P.E. Linda, 2010. Class break point determination using CK metrics thresholds. Global J. Comput. Sci. Technol., 10: 73-77.

Chhikara, A., R.S. Chhillar and S. Khatri, 2011. Evaluating the impact of different types of inheritance on the object oriented software metrics. Int. J. Enterprise Comput. Bus. Syst., Vol. 1.

Chidamber, S.R. and C.K. Kemerer, 1994. A metrics suite for object oriented design. IEEE Trans. Software Eng., 20: 476-493.

Dagpinar, M. and J.H. Jahnke, 2003. Predicting maintainability with object-oriented metrics: An empirical comparison. Proceedings of the 10th Working Conference on Reverse Engineering, November 13-16, 2003, Canada, pp: 155-164.

Dubey, S.K. and A. Rana, 2011. Assessment of maintainability metrics for object-oriented software system. ACM SIGSOFT Software Eng. Notes, 36: 1-7.

Emam, K.E., 2002. Object Oriented Metrics: A Review of Theory and Practice. In: Advances in Software Engineering, Comprehension, Evaluation and Evolution, Erdogmus, H. and O. Tanir (Eds.). Springer, New York, pp: 23-50.

Ferreira, K.A.M., M.A.S. Bigonha, R.S. Bigonha, L.F.O. Mendes and H.C. Almeida, 2012. Identifying thresholds for object-oriented software metrics. J. Syst. Software, 85: 244-257.

Harrison, R., S. Counsell and R. Nithi, 2000. Experimental assessment of the effect of inheritance on the maintainability of object-oriented systems. J. Syst. Software, 52: 173-179.

Herbold, S., J. Grabowski and S. Waack, 2011. Calculation and optimization of thresholds for sets of software metrics. Empirical Software Eng., 16: 812-841.

Knauf, R., S. Tsuruta and A.J. Gonzalez, 2007. Toward reducing human involvement in validation of knowledge-based systems. Trans. Syst. Man Cybernetics, 37: 120-131.

Knol, A.B., P. Slottje, J.P. van der Sluijs and E. Lebret, 2010. The use of expert elicitation in environmental health impact assessment: A seven step procedure. Environ. Health, Vol. 9. 10.1186/1476-069X-9-19

Krueger, T., T. Page, K. Hubacek, L. Smith and K. Hiscock, 2012. The role of expert opinion in environmental modelling. Environ. Modell. Software, 36: 4-18.

Kulkarni, R.B., 1985. Development of the performance prediction models using expert opinion. Proceedings of the 1st North American Pavement Management Conference, March 18-21, 1985, Toronto, Canada, pp: 135-146.

Leal, J., S. Wordsworth, R. Legood and E. Blair, 2007. Eliciting expert opinion for economic models: An applied example. Value Health, 10: 195-203.

Li, M. and C.S. Smidts, 2003. A ranking of software engineering measures based on experts opinion. IEEE Trans. Software Eng., 29: 811-824.

Lincke, R., J. Lundberg and W. Lowe, 2008. Comparing software metrics tools. Proceedings of the International Symposium on Software Testing and Analysis, July 20-24, 2008, Seattle, WA., USA., pp: 131-142.

Ma, B., C. Tan, Z.Z. Jiang and H. Deng, 2013. Intuitionistic fuzzy multicriteria group decision for evaluating and selecting information systems projects. Inform. Technol. J., 12: 2505-2511.

Mago, J. and P. Kaur, 2012. Analysis of quality of the design of the object oriented software using fuzzy logic. Proceedings of the International Conference on Recent Advances and Future Trends in Information Technology, April 2012, USA., pp: 21-25.

Misra, S.C. and V.C. Bhavsar, 2003. Relationships between selected software measures and latent bug-density: Guidelines for improving quality. In: Computational Science and its Applications, Kumar, V., M.L. Gavrilova, C.J.K. Tan and P. L'Ecuyer (Eds.). Springer, Berlin, Germany, pp: 724-732.

Mizuno, O. and H. Hata, 2010. An empirical comparison of fault-prone module detection approaches: Complexity metrics and text feature metrics. Proceedings of the 34th Annual Computer Software and Applications Conference, July 19-23, 2010, Seoul, South Korea, pp: 248-249.

Muketha, G.M., A.A.A. Ghani, M.H. Selamat and R. Atan, 2010. Complexity metrics for executable business processes. Inform. Technol. J., 9: 1317-1326.

Olague, H.M., L.H. Etzkorn, S. Gholston and S. Quattlebaum, 2007. Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. IEEE Trans. Software Eng., 33: 402-419.

Pai, G.J. and J.B. Dugan, 2007. Empirical analysis of software fault content and fault proneness using bayesian methods. IEEE Trans. Software Eng., 33: 675-686.

Pigoski, T.M., 2006. Practical Software Maintenance. John Wiley and Sons, New York.

Poornima, U.S., 2011. Unified design quality metric tool for object-oriented approach including other principles. Int. J. Comput. Appl., 26: 1-4.

Preece, A., 2001. Evaluating Verification and Validation Methods in Knowledge Engineering. In: Industrial Knowledge Management, Roy, R. (Ed.). Springer, London, pp: 91-104.

Rosenberg, L., R. Stapko and A. Gallo, 1999. Object oriented metrics and reliability. Proceedings of the 6th International Symposium on Software Metrics, November 4-6, 1999, USA., pp: 1-8.

Sharma, R., N. Budhija and B. Singh, 2012. Study of predicting fault prone software modules. Int. J. Adv. Res. Comput. Sci. Software Eng., Vol. 2.

Shatnawi, R., W. Li, J. Swain and T. Newman, 2010. Finding software metrics threshold values using ROC curves. J. Software Maint. Evol.: Res. Pract., 22: 1-16.

Singh, G., D. Singh and V. Singh, 2011a. A study of software metrics. Int. J. Comput. Eng. Manage., 11: 22-27.

Singh, P., K.D. Chaudhary and S. Verma, 2011b. An investigation of the relationships between software metrics and defects. Int. J. Comput. Appl., 28: 13-17.

Soi, I.M., 1985. Software complexity: An aid to software maintainability. Microelect. Reliab., 25: 223-228.

Spinellis, D., 2005. Tool writing: A forgotten art? IEEE Software, 22: 9-11.

Summerville, I., 2011. Software Engineering. 9th Edn., Pearson, Addison Wesley, New York.

Vivanco, R. and N. Pizzi, 2004. Finding effective software metrics to classify maintainability using a parallel genetic algorithm. Proceedings of the Genetic and Evolutionary Computation Conference, June 26-30, 2004, Seattle, WA, USA., pp: 1388-1399.

Wang, H., T.M. Khoshgoftaar and N. Seliya, 2011. How many software metrics should be selected for defect prediction? Proceedings of the 24th International Florida Artificial Intelligence Research Society Conference, May 18-20, 2011, Palm Beach, FL., USA., pp: 69-74.

Zhang, S., 2013. Optimizing the filling time and gate of the injection mold on plastic air intake manifold of engines. Inform. Technol. J., 12: 2473-2480.

Zhou, Y., B. Xu and H. Leung, 2010. On the ability of complexity metrics to predict fault-prone classes in object-oriented systems. J. Syst. Software, 83: 660-674.