

# Towards a Generic Model for Software Quality Prediction

Zeeshan Ali Rana  
LUMS, DHA  
Lahore, Pakistan  
zeeshanr@lums.edu.pk

Shafay Shamail  
LUMS, DHA  
Lahore, Pakistan  
sshmail@lums.edu.pk

Mian Muhammad Awais  
LUMS, DHA  
Lahore, Pakistan  
awais@lums.edu.pk

## ABSTRACT

Various models and techniques have been proposed and applied in literature for software quality prediction. Specificity of each suggested model is one of the impediments in development of a generic model. A few models have been quality factor specific whereas others are software development paradigm specific. The models can even be company specific or domain specific. The amount of work done for software quality prediction compels the researchers to get benefit from the existing models and develop a relatively generic model. Development of a generic model will facilitate the quality managers by letting them focus on how to improve the quality instead of employing time on deciding which technique best suites their scenario. This paper suggests a generic model which takes software as input and predicts a quality factor value using existing models. This approach captures the specificity of existing models in various dimensions (like quality factor, software development paradigm, and software development life cycle phase etc.), and calculates quality factor value based on the model with higher accuracy. Application of the model has been discussed with the help of an example.

## Categories and Subject Descriptors

D.2.8 [Software Engineering]: Metrics—*software science, product metrics*

## General Terms

Measurement

## Keywords

Prediction, Generic models, Measurement-based prediction, Metrics

## 1. INTRODUCTION

Software quality prediction helps minimize software costs by allowing the mitigation of risks in early stages of software

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WoSQ'08, May 10, 2008, Leipzig, Germany.

Copyright 2008 ACM 978-1-60558-023-4/08/05 ...\$5.00.

development process [4, 15, 8]. It further helps in preparation of better resource allocation plans [33, 29] and test plans [33, 13, 23, 19]. These factors help produce a good quality software which results in satisfied customer and healthier return on investment. Various organizations and public departments have been involved in studies related to quality prediction like Commission of the European Communities' Strategic program for Research in Information Technology [1], Northern Telecom Limited, USA [15], Nortel, USA [14], NASA [18], National Natural Science Foundation of China [31] are a few examples. Earlier studies [13, 20, 33, 21, 22] have viewed quality in different aspects and have been limited to a particular quality factor (for example reliability, maintainability [5]). Moreover the application of those models is very context specific [28, 24]. Details of all these techniques can be found in [24]. Specificity of these approaches is a limitation of software quality prediction study and refrains the community from taking full benefit of the existing work. That is why a very small number of suggested models are used in industry [3]. A generic model based on existing models will help quality prediction endeavor. This paper suggests such a model which is composed of existing models (called component models). Given a software in terms of its collected metrics, the model selects the most appropriate applicable model to predict the desired quality factor value. The proposed model can be extended for as many quality factors as there are models available for. Like an integrated approach suggested by Wagner et al. [28], the proposed model also requires the prior analysis of the important quality factors. Furthermore, it will automate the selection of an appropriate prediction model.

Rest of the paper is organized as follows: In section 2 we discuss the related approaches towards generic models. We present our approach in section 3 and use an example to explain the suggested approach in section 4. In section 5 we discuss the limitations and issues with the suggested approach before concluding the paper in section 6.

## 2. RELATED WORK

Various studies on product-based quality prediction models have been done [7, 9, 13, 11, 25, 23]. Ganesan et al. employed case-based reasoning to predict design faults [7]. Grosser et al. [9] suggested a technique which was suitable for object oriented (OO) software only. Company specific [17] and domain specific ([12] for telecommunication systems) studies have also been presented. Because the work has been done in various dimensions, need of a generic model for software quality has been felt [3, 27, 28, 30]. But the

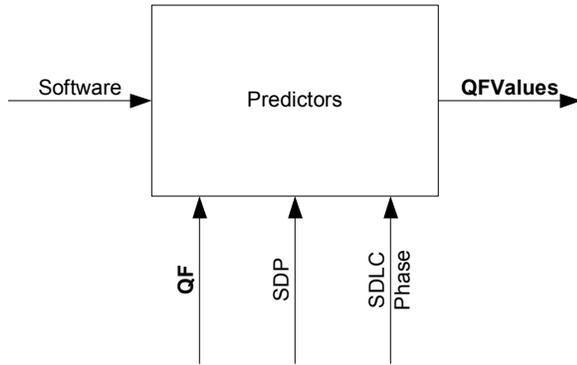


Figure 1: Block diagram of suggested model.

application and context specific nature of existing models causes difficulty in taking full advantage of the existing work. Fenton et al. [6] have presented a critique of existing models and highlighted their weaknesses [6]. A few models, generic for a certain quality factor such as usability [30], have been suggested. Bouktif et al. [3] have considered the unavailability of large data repositories as an obstacle to generalize, validate and reuse existing models. They have suggested a technique for selecting an appropriate model from a set of existing models. Their approach reused existing models but it was restricted to selection of an appropriate model only. Their major focus was on facilitating a company in adapting object oriented software quality predictors to a particular context. Though Wagner [27] has suggested an approach to reduce the effort to apply a prediction model, some other issues yet need to be addressed. It is difficult to avoid specific behavior of a predictor of software quality. To address this issue models have been divided in different dimensions of specificity. Wagner et al. [28] have argued that software quality models differ along six dimensions namely purpose, view, attribute, phase, technique and abstraction. Rana et al. [24] have identified four dimensions for quality prediction models: software development paradigm (SDP), software development lifecycle (SDLC) phase, quality factor and approach of the model. Both [28] and [24] have two dimensions in common which are attribute (quality factor) and phase (SDLC phase). Dimension SDP [24] is an important information to capture regarding a model.

Other issues which hinder the development of a generic model are inconsistent names of software metrics, and non-uniform input to the existing models. For example model by Guo et al. [10] refers to lines of code as TC (Total Code lines) whereas many other models (like [13]) refer to it as LOC. On the other hand, models by Khoshgoftaar et al. [15, 14] refer to TC as ‘Total calls to other modules’. Such inconsistencies make the existing models incomparable. Rana et al. [24] have done unification of metrics nomenclature. Their work paves the way towards development of a generic model.

### 3. OUR APPROACH

The block diagram of suggested prediction model is shown in figure 1. It outputs a set of quality factor values, **QF-Value**, which can be expressed by the following expression:

$$\mathbf{QFValue} = f(\text{Software}, \mathbf{QF}, \text{SDP}, \text{SDLCPhase}) \quad (1)$$

where *SDLCPhase* is the Software Development Life Cy-

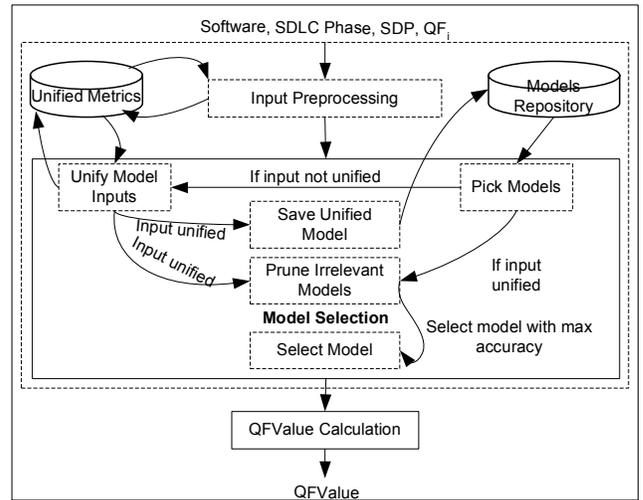


Figure 2: Model Architecture.

Table 1: A snapshot from Unified Metrics Database

Metric	Unified Name	Aliases
Lines of code	LOC	TC LOC
Total Comments	TCOMM	COM TCOMM
Cohesion Metric	CohM	COM CohM
Depth of Inheritance Tree of a Class	DIT	DEPTH DIT

cle (SDLC) phase whose collected metrics are to be used, *SDP* is the software development paradigm and **QF** is set of quality factors which need to be determined. The software is provided in terms of calculated metrics. The model has three major phases as shown in figure 2:

- Input Preprocessing
- Model Selection
- QF Calculation

**Input Preprocessing:** This phase handles the inconsistencies of input metrics’ names with the unified metrics database. The unified metrics database contains the data structures as shown in table 1 and is based on unifications done in [24]. This phase is carried out manually and it is needed so that appropriate models can be selected for this software.

**Model Selection:** Model selection phase has further five sub-activities. Each model is checked if its input interface has been unified or not. If the input interface is not already unified then ‘Unify Model Inputs’ sub-activity is carried out. Otherwise the search for relevant models is started. For example for a few models shown in figure 3 model Y and Z are relevant models to predict number of errors in object

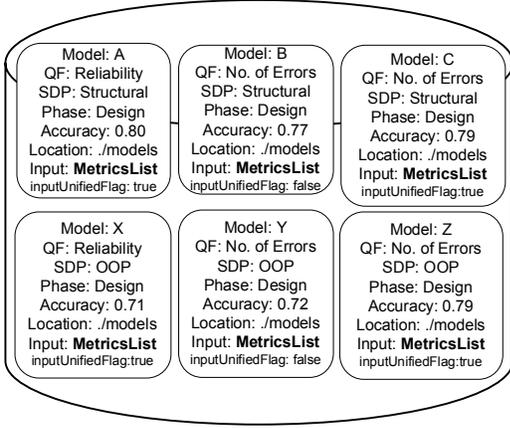


Figure 3: Model Repository.

Table 2: Dimensions Captured

Dimension	Example
Purpose	constructive, predictive
View	product-based, value-based
Attribute	reliability, number of errors
Phase	design, implementation, testing
Technique	model based on testing process
Abstraction	general, context specific
SDP	structural, object oriented

oriented paradigm. The search is narrowed down through pruning the irrelevant models. The process is made clear in section 4 with an example. Once the list of applicable models has been prepared, it is passed to the ‘Select Model’ process which then selects a model using the expression

$$Max(Acc_1, Acc_2, \dots, Acc_i, \dots, Acc_m) \quad (2)$$

where  $m$  is total number of models for a certain type of software and quality factor and  $Acc_i$  is accuracy of the  $i^{th}$  model for that type of software and quality factor. For example, in figure 3 model A is the selected predictor of reliability for software developed using structural programming.

**QF Calculation:** This phase invokes the prediction model selected in previous phase and inputs the software metrics to that model. The output of the invoked model becomes output of our model.

### 3.1 Models Repository

The suggested model captures specificity of component models along the seven dimensions listed in table 2. Specificity information regarding each component model is captured when it is placed in the repository shown in figure 3. A model with given accuracy  $\alpha$  or above can be put in the repository, where accuracy is defined as percentage of the correct predictions when predictor was run on test data. In our case  $\alpha = 65\%$ .

### 3.2 Issues with Component Models’ Inputs

Issues regarding sufficient number of inputs have been noticed. For example one model calculates a certain quality

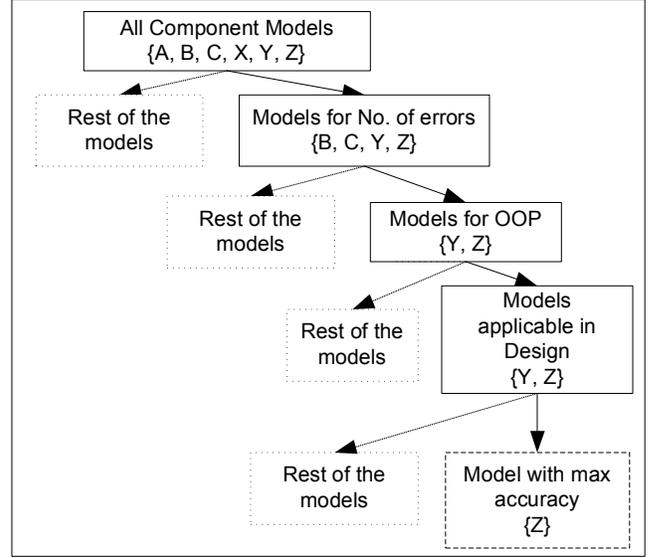


Figure 4: Models Pruning and Selection on an example.

factor with the help of 10 metrics whereas another model needs 12 metrics to do the same job. Assume there is a model A which has 90% accuracy for object oriented software and it needs the metric named *class hierarchy* to predict *stability*. But this metric is not a part of the input software. Now there is another model B with 85% accuracy but all the required metrics are available. In such a case our selector will be comparing model B with other models which satisfy the constraint and will not consider model A for comparison.

## 4. AN EXAMPLE

In this section we elaborate the working of our model using dataset kc1 [2]. The dataset originally comprises of 94 attributes which were used by Koru et al. for study in quality prediction [16]. On the basis of other studies on object oriented metrics [32, 26], we have selected eight significant metrics for this study shown in table 3. Suppose that we have the models shown in figure 3. Let Y be a model based on Linear Regression (LR) and Z be a Support Vector Regression (SVR) based model. Their accuracies (calculated using first 100 instances of the dataset kc1) are 69% and 83% respectively. Rest of the information regarding the model remains as it is in figure 3. Now our model is provided with the following information in addition to the list of software whose quality factor value is to be determined:

*SDLCPhase = Design*  
*SDP = ObjectOriented*  
*QF = NumberofErrors*

As shown in figure 2 the first step is ‘Input Preprocessing’. At this step unification of input metrics is done by using ‘Unified Metrics’ database. The unified names are shown in third column of table 3. In the next step the inputs of all models are unified. For example in our case, there are two models Y and B as shown in figure 3 which need their

**Table 3: Object-Oriented Metrics Used**

Metric	Name in dataset	Unified Name
Coupling Between Objects	COUPLING_BETWEEN_OBJECTS	CBO
Depth of Inheritance Tree of a class	DEPTH	DIT
Lack of Cohesion in Methods	LACK_OF_COHESION_OF_METHODS	LCOM
Number of Children	NUM_OF_CHILDREN	NOC
Dependence on an a descendant	DEP_ON_CHILD	DEP_ON_CHILD
Count of calls by higher modules	FAN_IN	FAN_IN
Response For Class	RESPONSE_FOR_CLASS	RFC
Weighted Methods per Class	WEIGHTED_METHODS_PER_CLASS	WMC

**Table 4: Summary of Results**

	Model Z (SVR)	Model Y (LR)
MAE	4.4669	5.5415
RAE	73.9543%	91.7451%
RMSE	10.6442	10.5929
MAE	97.4832%	97.0131

inputs to be unified. We ‘Unify Model Inputs’ and store back the model in the repository with *inputUnifiedFlag* set to *true*. The next step is to ‘Prune Irrelevant Models’. Figure 4 shows that only models Y and Z are relevant models and the model Z having higher accuracy is selected to predict the *numberoferrors*. Once the best model is selected, the next step is to calculate the QF value for all input software. We applied model Z (SVR) on the dataset kc1 [2] to predict *number of errors* in the 145 software. We then applied model Y (LR) to the same dataset in order to validate our selection of model on the basis of accuracy. Summary of the results is shown in table 4 where we observed four types of errors namely Mean Absolute Error (MAE), Relative Absolute Error (RAE), Root Mean Squared Error (RMSE) and Root Relative Squared Error (RRSE). The table shows that the error values for model Z are smaller than the error values for model Y. The lower values of MAE and RAE imply that values predicted by model Z were closer to actual values.

## 5. DISCUSSION

Currently the proposed model is handling structural and object oriented development paradigms only, but it is extendible to other paradigms. It can also be extended for as many quality factors as there have been models available for. The model is usable at any stage of software life cycle provided the models repository contains models applicable in that phase. Our model caters for the specificity of a component predictor by taking three control inputs (Quality Factor, Software Development Life Cycle Phase, and Software Development Paradigm), which contribute towards selection of a predictor. We are using the product-based existing models unlike Bouktif et al. [3] who are using classification based models.

**Accuracy as selection criterion:** Any model can be plugged into the suggested model provided its accuracy does not fall below the minimum threshold  $\alpha$ . Li et al.[17] have argued that accuracy is not a good criterion for model selection. They have asserted that in context of product testing

prioritization, model selection should be done considering the specificity of the predictors. Although this concern applies to our selection criteria as well but we are restricting ourselves to accuracy for the time being.

**Generality:** Integrated approach to predict software quality by Wagner et al. [28] is a significant contribution since it provides the ways to describe generic quality characteristics and how to adapt to different contexts. Our paper suggests a model which is more general than the integrated approach by Wagner et al. [28] since it does not require each organization to develop a base model before including purpose models. The purpose models are specific models which are derived from the base model in the light of quality goals [28]. Our idea is to have a predictor which requires minimum effort from quality engineer in predicting quality of software. As shown in section 4, our model only needs the software metrics and will do the rest of the work including model selection and QF calculation.

## 6. CONCLUSIONS

In this paper we have suggested an approach which takes benefits from the existing work. The approach caters for the specific nature of component models and is extendible as well as scalable. Extendible in the sense that it can be used for more than one paradigms as well as for different SDLC phases. Moreover it can have the capability to predict more than one quality factors. A new component model for another quality factor can be added to the generic framework at any time. The only restriction on the inclusion of a component model is that the accuracy of the model can not fall below a threshold  $\alpha$  which is set to 65% in this paper. The paper shows working of the model through an example using 145 data instances. The example depicts that the model selected based on its accuracy was better selection in this case. The paper identifies some issues which still need to be addressed. The first issue is whether a software should be provided in the form of metrics or as source code. The suggested approach took software in the form of metrics but faced problems in using existing models as discussed in section 3.2. So taking software in the form of metrics can affect the accuracy of the prediction. In the second case the suggested model first needs to calculate the software metrics of its choice. This might be time consuming but while predicting software quality, accuracy is the most important factor. So time taken to build and use the model can be compromised as long as it guarantees higher accuracy. In future our plan is to explore the tradeoffs between selection of the above mentioned two choices.

## 7. ACKNOWLEDGMENTS

We would like to thank Malik Jahan Khan and Numan Sheikh for their support during this work. We also thank Higher Education Commission (HEC) of Pakistan and Lahore University of Management Sciences (LUMS) for funding this research.

## 8. REFERENCES

- [1] S. Abe, O. Mizuno, T. Kikuno, N. Kikuchi, and M. Hirayama. Estimation of project success using bayesian classifier. In *Proceedings of The 28th International Conference on Software Engineering, ICSE'06*, 2006.
- [2] G. Boetticher, T. Menzies, and T. Ostrand. Promise repository of empirical software engineering data, 2007.
- [3] S. Bouktif, D. Azar, D. Precup, H. Sahraoui, and B. Kt'egl. Improving rule set based software quality prediction: A genetic algorithm-based approach. *Journal of Object Technology*, 3(4):227–241, April 2004.
- [4] L. C. Briand, V. R. Basili, and C. J. Hetmanski. Developing interpretable models with optimized set reduction for identifying high-risk software components. *IEEE Transactions on Software Engineering*, Vol. 19(No. 11):1028–1044, November 1993.
- [5] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J.-F. Girard. An activity-based quality model for maintainability. In *Proceedings of the 23rd International Conference on Software Maintenance (ICSM 2007)*. IEEE Computer Society Press, 2007.
- [6] N. E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, Vol. 25(No. 5):675–687, September/October 1999.
- [7] K. Ganesan, T. M. Khosgoftaar, and E. B. Allen. Case-based software quality prediction. *International Journal of Software Engineering and Knowledge Engineering*, Vol. 10(No. 2):139–152, February 2000.
- [8] S. S. Gokhale and M. R. Lyu. Regression tree modeling for the prediction of software quality. In *Proceedings of The 3rd ISSAT Intl. Conference on Reliability*, 1997.
- [9] D. Grosser, H. A. Sahraoui, and P. Valtchev. Analogy-based software quality prediction. In *7th Workshop On Quantitative Approaches In Object-Oriented Software Engineering, QAOOSE'03*, June 2003.
- [10] P. Guo and M. R. Lyu. Software quality prediction using mixture models with em algorithm. In *Proceedings of The First Asia-Pacific Conference on Quality Software*, 2000.
- [11] H. A. Jensen and K. Vairavan. An experimental study of software metrics for real-time software. *IEEE Transactions on Software Engineering*, Vol. SE-11(No. 2):231–234, February 1985.
- [12] T. M. Khosgoftaar, D. L. Lanning, and A. S. . Pandya. A comparative study of pattern recognition techniques for quality evaluation of telecommunications software. *IEEE Journal On Selected Areas In Communications*, Vol. 12(No. 2):279–291, February 1994.
- [13] T. M. Khosgoftaar and J. C. Munson. Predicting software development errors using software complexity metrics. *IEEE Journal On Selected Areas In Communications*, Vol. 8(No. 2), February 1990.
- [14] T. M. Khosgoftaar and E. B. Allen. Predicting fault-prone software modules in embedded systems with classification trees. In *Proceedings of The 4th IEEE International Symposium on High-Assurance Systems Engineering*. IEEE, Computer Society, 1999.
- [15] T. M. Khosgoftaar, E. B. Allen, K. S. Kalaichelvan, and N. Goel. Early quality prediction: A case study in telecommunications. *IEEE Software*, 1996.
- [16] A. G. Koru and H. Liu. An investigation of the effect of module size on defect prediction using static measures. In *Proceedings of International Workshop on Predictor Models in Software Engineering (PROMISE '05)*. ACM Press, 2005.
- [17] P. L. Li, J. Herbsleb, M. Shaw, and B. Robinson. Experiences and results from initiating field defect prediction and product test prioritization efforts at abb inc. In *Proceedings of The 28th International Conference on Software Engineering, ICSE'06*, 2006.
- [18] A. Mockus, P. Zhang, and P. L. Li. Predictors of customer perceived software quality. In *Proceedings of The 27th International Conference on Software Engineering, ICSE'05*, 15-21 May 2005.
- [19] S. N. Mohanty. Models and measurements for quality assessment of software. *ACM Computing Surveys*, 11(3):251–275, September 1979.
- [20] J. C. Munson and T. M. Khosgoftaar. The detection of fault-prone programs. *IEEE Transactions on Software Engineering*, Vol. 18(No. 5):423–434, May 1992.
- [21] N. Nagappan and T. Ball. Use of relative code churn measures to predict system defect density. In *Proceedings of The 27th International Conference on Software Engineering, ICSE'05*, 2005.
- [22] N. Nagappan, T. Ball, and A. Zeller. Mining metrics to predict component failures. In *Proceedings of The 28th International Conference on Software Engineering, ICSE'06*, 2006.
- [23] L. M. Ottenstein. Quantitative estimates of debugging requirements. *IEEE Transactions on Software Engineering*, Vol. SE-5(No. 5):504–514, September 1979.
- [24] Z. A. Rana, S. Shamail, and M. M. Awais. A survey of measurement-based software quality prediction techniques. Technical report, Lahore University of Management Sciences, December 2007.
- [25] V. Schneider. Some experimental estimators for developmental and delivered errors in software development projects. *ACM SIGMETRICS Performance Evaluation Review*, Vol. 10(Issue 1):169–172, 1981.
- [26] M.-H. Tang, M.-H. Kao, and M.-H. Chen. An empirical study on object-oriented metrics. In *Proceedings of Sixth International Software Metrics Symposium 1999*, pages 242–249. IEEE, 1999.
- [27] S. Wagner. Global sensitivity analysis of predictor models in software engineering. In *Proceedings of 3rd International Workshop on Predictor Models in Software Engineering (PROMISE '07)*. IEEE Computer Society Press, 2007.

- [28] S. Wagner and F. Deissenboeck. An integrated approach to quality modelling. In *Proceedings of 5th Workshop on Software Quality (5-WoSQ)*. IEEE Computer Society Press, 2007.
- [29] Q. Wang, B. Yu, and J. Zhu. Extract rules from software quality prediction model based on neural network. In *Proceedings of The 16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI'04*, 2004.
- [30] S. Winter, S. Wagner, and F. Deissenboeck. A comprehensive model of usability. In *Proceedings of Engineering Interactive Systems 2007*. Springer-Verlag, 2007.
- [31] F. Xing, P. Guo, and M. R. Lyu. A novel method for early software quality prediction based on support vector machine. In *Proceedings of The 16th IEEE International Symposium on Software Reliability Engineering*. IEEE, 2005.
- [32] P. Yu, T. Systs, and H. Muller. Predicting fault-proneness using oo metrics an industrial case study. In *Proceedings of Sixth European Conference on Software Maintenance and Reengineering (CSMR.02)*. IEEE Computer Society, 2002.
- [33] X. Yuan, T. M. Khoshgoftaar, E. B. Allen, and K. Ganesan. An application of fuzzy clustering to software quality prediction. In *Proceedings of The 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*. IEEE, 2000.