Machine Learning-based Software Quality Prediction Models: State of the Art

Hamdi A. Al-Jamimi and Moataz Ahmed
Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia
{aljamimi, moataz}@kfupm.edu.sa

***Abstract*—Quantification of parameters affecting the software quality is one of the important aspects of research in the field of software engineering. In this paper, we present a comprehensive literature survey of prominent quality molding studies. The survey addresses two views: (1) quantification of parameters affecting the software quality; and (2) using machine learning techniques in predicting the software quality. The paper concludes that, model transparency is a common shortcoming to all the surveyed studies.**

Keywords— Software quality; quality assessment; prediction model; machine learning, transparency

I. INTRODUCTION

Software quality means to which degree a system, or process satisfies identified requirements [1]. Indeed, the quality describes and measures what a system does, as well as it describes how the system is built and performs. According to ISO 9126 standard definition, software quality is "*the totality of features and attributes of a software product that bear on its ability to satisfy stated or implied needs*"[2]. Software features and attributes can be categorized into two different groups: internal and external. The internal quality attributes can be measured directly during the different stages of the software development lifecycle as well as after delivery, while the external quality attribute can be seen as multidimensional in the way many quality factors are connected to other quality factors, and each depends on a number of internal quality attributes[3].

A software quality model can be used to specify various external attributes that are of interest to the stakeholders and their level of importance. It also depicts the functional relationship between the external attributes of interest and the corresponding internal attributes which we can measure. Rather than using quality models to assess external attributes after the fact; it is more common to use quality models to predict external quality attributes before it is too late. In other words, the quality model allows one to predict the external quality attributes as a function of several other variables which are indeed measureable internal quality attributes. In this case, such internal attributes serve as predictors to predict future external quality attributes at early stages during the development lifecycle. For example, if maintainability is an external quality of interest, the software designer would be interested in predicting the maintainability of his/her design from some internal attributes (e.g., coupling and cohesion) that are measurable during the design process rather than waiting to assess maintenance effort and cost after the fact.

Two major tasks involved when assessing/predicting a particular external quality attribute: identifying the factors or the internal quality attributes contributing to the intended external attribute; and identifying the functional relationship between the external attribute and the corresponding internal attributes. Different prominent generic software quality models were proposed by various researchers such as, Boehm's Model [4], McCall Model [5], ISO/IEC 9126 model [6] and Dromey's Model [7]. These models were developed to allow defining and measuring the software quality in different perspectives by describing and evaluating a set of software attributes. However, these prominent models provide more of frameworks for assessing and predicting the software quality than concrete models ready for practitioners to apply. In particular, these quality models do not provide specific functional relationships between the internal and external attributes. It is worth noting here that in most cases, such relationships could be more complex than just a simple linear relationship. Guided by these prominent models, many attempts have been proposed in the literature to mainly address the functional relationships. In this paper, we present a comprehensive survey that addresses two views of these attempts. While Section II introduces quantification of parameters affecting the software quality; Section III surveys the studies that utilize machine learning (ML) techniques in predicting the software quality. Then, the open issues and shortcomings of the surveyed studies are discussed in Section IV.

II. QUALITY PARAMETERS QUANTIFICATION

Currently, quantification of parameters affecting software quality is an important filed of research in software engineering (SWE). Different parameters are quantified in various studies due to their effects on the software quality. The software quality models can be categorized into fixed -models and define-your-own models [8].

A. The fixed-model approach

This type of models specifies a particular set of qualities where the set of attributes indentified by the customer is a

978-1-4799-0604-8/13/$31.00 ©2013 IEEE

subset in the predefined model. In order to measure and control a particular quality attribute, there is a need to use the sub-characteristics, the measures, and the relationships associated with the fixed model.

Boehm's model [4] was defined to provide a set of "well defined, well differentiated characteristics of software quality". This model is viewed as a hierarchical model, where the different quality criteria in the model are subdivided as the hierarchy of the model is extended. Two levels of the quality criteria are identified, such that the intermediate level is further divided into measurable characteristics. McCall model [5] was aimed at system developers to be used during the development process. The model identifies three areas of software work: product operation, product revision and product transition. It reflects developers' priorities as well as users' views. Dromey [7] proposed a model for software product quality. The model establishes the link between product characteristics and less tangible quality attributes. That is, the illustrated model provides an explicit process for building quality-carrying properties into software. In turn, these properties imply specific quality attributes. Moreover, the model can help in conducting a systematic search for quality defects in software.

Lamouchi et al. [9] quantified the software quality factors by a hierarchical model. The factors are subdivided into criteria and sub criteria at various levels. The last level quantified different software metrics affecting the different factors. Srivastava et al. [10] quantified the parameters of the software quality through different perspectives: developer's, project manager's, and user's perspectives. They considered the weighted average for different factors to get the actual software quality. Kanellopoulos et al. [11] evaluated the source code quality and static behavior of a software system, based on the standard ISO/IEC-9126 [6], with the help of Analytical Hierarchy Process (AHP) model

B. The define-your-own-model approach

This approach contrasts the fixed models approach in the sense that no specific quality attributes are defined. On the other hand, a consensus on relevant quality attributes is recognized for a specific system in cooperation with the user. Then, the defined attributes are decomposed to quality characteristics that can be measured and their metrics. The decomposition can be guided by an existing quality model. The relationships between quality attributes and characteristics could then be defined in two ways. First, by directly-defined model which defined by project stakeholders. Second, by indirectly-defined model that can be generated automatically.

The directly-defined models- are viewed as dependency graphs. Examples of such approaches introduced in [12, 13]. Samoladas et al. [12] presented a hierarchical quality model to evaluate the source code as well as the community processes. Lazić et al. [13] proposed a model that can be utilized to trace the design decisions as well as the potential alternatives. This can help reduce the cost when evaluating different design alternatives and switching between them.

Indirectly-defined models- come originally from two main domains, namely artificial intelligence and mathematics. The quality model could be influenced by selected technique and its parameters. Nevertheless, the used technique has no direct impact on the output quality model. Indeed, the represented quality relationships in such models are complicated which affect negatively the project stakeholder understanding.

III. MACHINE LEARNING-BASED SOFTWARE QUALITY PREDICTION

ML techniques have been utilized in many different problem domains as this field concentrates on building algorithms that have the ability to enhance their performance automatically by experience. Applying ML techniques to SWE produced promising and encouraging results [14].

Support vector machine (SVM) has been applied successfully in many SWE prediction models. SVM has been employed for the maintenance effort predication [15]. Moreover, different studies have utilized SVM for predicting the software fault-proneness modules [16-18].

Bayesian network (BN) has been used in various studies in SWE area due its ability to integrate both empirical data and expert opinions. Some studies have been focused on software quality prediction [19] and development effort [20]. Wagner [21] proposed a framework for building BN for predicting software quality using the activity-based quality models. Similarly, Radliński [22] proposed a BN model for integrated software quality prediction. Then the author enhanced the proposed framework again using BN [23]. Other studies utilized BN to assess and predict the maintainability of the software [24].

Kanmani et al. [25] introduced the use of neural networks (NN) as a tool for predicting the software faults. Yang et al. [26] also proposed a software quality prediction model based on a fuzzy NN identify design errors in software products in the early stages of a software lifecycle. Moreover, NN have been utilized to predict early a specific quality attributes for example, reliability [27] and maintenance effort [28].

Since the relationship between the internal and external quality attributes is surrounded with impression and uncertainty, different studies in the literature have made attempts to utilize the capability of fuzzy logic (FL) to estimate the software quality. Mittal et al. [29] proposed a FL based approach to quantify the quality of software, where the examined software has been given quality grades on the basis of two metrics. Srivastava et al. [30] tried to rank the software quality using the fuzzy multi criteria approach. The approach ranked the software based on software requirement specifications documents. They accomplished a similar analysis using the ISO/IEC 9126 quality model [31]. Yang [32] proposed an approach to measure software product quality with ISO standards based on FL technique. A Fuzzy AHP model was proposed by Yuen et al. [33] to evaluate the software quality and to select software vendor under uncertainty. The model ranks the different software to give an opportunity to select the best one appropriately. In additional work [34] they employed Fuzzy AHP and specifically fuzzy logarithmic least square method to estimate the software quality. Mago et al. [35] used FL to analyze the quality of object oriented software design.

Various studies have been focusing on predicting the maintainability of software. Many FL-based approaches have been proposed to measure the quality attributes after the fact. For instance, Mittal et al. [36] proposed a FL-based approach to assess the maintenance productivity of software system. Sharam et al. [37] introduced FL-based approach for prediction of maintainability of component-based systems. In addition, another FL-based approach was proposed by Singh et al. [38] to predict the software maintenance. In our previous work [39] we presented initial experiments for building FL-based models to predict software maintainability. Other researchers considered different ML techniques to predict the software maintainability. Similarly, different FL approaches have been introduced to assess different quality attributes such as usability [40], understandability [41], and reusability [42, 43], and reliability [44].

It is worth noting here that none of the attempts discussed above considered the model transparency as an objective. To the best of our knowledge, previously proposed models were not transparent enough for human to incorporate their knowledge.

IV. Discussion and Open Issues

Model transparency refers to the model property where the functional relationships are interpretable by human; it allows incorporating two kinds of knowledge: expert knowledge and hidden knowledge in data. Experts would be able to directly manipulate the model structure for any necessary additions or modifications [45, 46].

Transparency poses an important constraint on quality models to be effective. This is because experts' input offers an important portion of information when it comes to building quality prediction models; historical data provide the other portion. Historical data provide numerical quantitative measurements from past projects regarding the internal and external quality attributes. Human experts use their experience to provide qualitative descriptions of the correlation between the internal and external quality attributes. Experts use linguistic values, e.g., high, medium, and low, which are imprecise by nature. Same applies to relationships suggested by experts; they are also imprecise.

In this regard, FL offers important features over other approaches because of its ability to naturally represent the expert-provided qualitative linguistic knowledge, accommodate imprecise data, and apply flexible inference rules. As for transparency, although an implicit assumption states that naturally fuzzy rules can be interpreted easily, this couldn't be right all the time. The assumption could be wrong when the generated partitioning is meaningless for experts or when dealing with complex multivariable systems [45]. An example of the lack of the transparency, when using fuzzy logic model to predict particular software attribute. Different fuzzy sets are used, where those sets should be shared across all rules. Recall that the fuzzy sets can be generated from data or given by experts. However, the resultant system of the training procedure suffers from a shortcoming that hurts transparency. This is mainly because the training process changes the number of linguistic values possible for each linguistic variable. Mainly, the training procedure increases the number linguistic values as the number of rules increases.

To get a clear idea of the issue, let's get an insight into the physical representation of a fuzzy model. Assume we use a model with three input variables (X, Y and Z), where two different linguistic values (fuzzy sets) have been defined for each input variable. The rules represent the different linguistic values for each variable, where each combination of the linguistic values has its own response. That is, the input variable X will be represented in each rule either by x_1 or x_2 and the same for the variables Y and Z. When initializing the rules it is clear for the experts the meaning of the used linguistic variables. However, after training the fuzzy model it would not end up with the same fuzzy sets for each input variable (i.e., two sets for each). The fuzzy sets of variable X may be as the same number of the number of the rules (e.g. if number if rules is n, then the fuzzy sets of X are x_1, x_2, x_3,…,x_n) which is meaningless for the expert.

In view of that, we study the impact of the automatic rule generation and structure optimization on model transparency [47]. We present a generic process for building fuzzy-based quality prediction models. The process is meat to serve as quality assessment framework that practitioners can use according to the data available. A case study was conducted using Maintainability as an external attribute of interest. Our long term objective is to conduct similar experiments in the future on other external attributes as well whenever data is available; and come up with a general quality model.

V. Conclusion

In this paper, we conducted a comprehensive literature survey of current software quality prediction modeling approaches. Our goal is to identify the state of the art with regard to utilizing the two major sources of knowledge for building models. Pre-requisite to effective combination of knowledge sources is the transparency of the model. The survey revealed that none of the current models address the model transparency issue.

Acknowledgment

The authors wish to acknowledge King Fahd University of Petroleum and Minerals for supporting this research.

References

[1] Gillies and A. C, Software Quality: Theory and management, 1992, London: Chapman and Hall.

[2] Standard, I.I., ISO-9126 Software Product Evaluation - Quality Characteristics and Guidelines for Their Use, 1991.

[3] Fenton, N.E. and S.L. Pfleeger, Software Metrics: A Rigorous and Practical Approach1998, Boston, MA, USA PWS Publishing Co.

[4] Boehm, B.W., et al., Characteristics of Software Quality, 1978: North Holland Publishing Company.

[5] McCall, J.A., P.K. Richards, and G.F. Walters, Factors in Software Quality I. Vol. I, and III, US Rome Air Development Center Reports - NTIS AD/A-049 014, NTIS AD/A-049 015 and NTIS AD/A-049 016, U. S. Department of Commerce., Editor 1977.

[6] Standard, I.I.I., Software Engineering - Product quality - Part 1: Quality model. 2001.

[7] Dromey, R.G., A Model for Software Product Quality. IEEE Transactions on Software Engineering, 1995. 21(146-162).

[8] Trendowicz, A. and T. Punter. Quality Modeling for Software Product Lines. in In: 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'03). 2003.

[9] Lamouchi, O., A.R. Cherif, and N. Lévy. A framework based measurements for evaluating an IS quality. in Proceedings of the fifth on Asia-Pacific conference on conceptual modelling. 2008. Wollongong, NSW, Australia.

[10] Srivastava, P.R. and K. Kumar. An Approach towards Software Quality Assessment. in Communications in Computer and Information Systems Series (CCIS Springer Verlag). 2009.

[11] Kanellopoulos, Y., et al., Code Quality Evaluation Methodology Using The Iso/Iec 9126 Standard. International Journal of Software Engineering & Applications (IJSEA), 2010. 1(3): p. 17-36.

[12] Samoladas, I., et al., The SQO-OSS quality model: measurement based open source software evaluation. 2008.

[13] Lazić, L., A. Kolašinac, and D. Avdić, The software quality economics model for software project optimization. WSEAS transactions on computers, 2009. 8(1): p. 21-47

[14] Zhang, D. and J.J.P. Tsai, Machine Learning and Software Engineering. Software Quality Journal, 2003. 11(2): p. 87-119.

[15] Jin, C. and J.-A. Liu. Applications of Support Vector Mathine and Unsupervised Learning for Predicting Maintainability using Object-Oriented Metrics. in The Second International Conference on MultiMedia and Information Technology. 2010

[16] Singh, Y., A. Kaur, and R. Malhotra. Software Fault Proneness Prediction Using Support Vector Machines. in Proceedings of the World Congress on Engineering. 2009. London, U.K.

[17] Malhotra, R., A. Kaur, and Y. Singh, Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines. Int J Syst Assur Eng Manag (July-Sept), 2010. 1(3): p. 269-281.

[18] Al-Jamimi, H. and L. Ghouti. Efficient prediction of software fault proneness modules using support vector machines and probabilistic neural networks. in 5th Malaysian Conference in Software Engineering (MySEC). 2011.

[19] Radliński, Ł., A conceptual Bayesian net model for integrated software quality prediction Journal of Annales UMCS Informatica, 2011. 11(4): p. 49-60.

[20] Radlinski, L., A Survey of Bayesian Net Models for Software Development Effort Prediction. International Journal of Software Eng. and Computing, 2010. 2: p. 95-109.

[21] Wagner, S. A Bayesian Network Approach to Assess and Predict Software Quality Using Activity-Based Quality Models. in In: 5th Int. Conf. on Predictor Models in Software Engineering. 2009. ACM Press, New York.

[22] Radliński, Ł. A Framework for Integrated Software Quality Prediction Using Bayesian Nets. in ICCSA 2011. 2011.

[23] Radliński, Ł. Enhancing Bayesian Network Model for Integrated Software Quality Prediction. in The Fourth International Conference on Information, Process, and Knowledge Management. 2012.

[24] Jeet, K. and R. Dhir, Bayesian and Fuzzy Approach to Assess and Predict the Maintainability of Software: A Comparative Study. International Scholarly Research Network ISRN Software Engineering, 2012.

[25] Kanmani, S., et al., Object-oriented software fault prediction using neural networks. Information and Software Technology, 2007. 49(5): p. 483-492.

[26] Yang, B., L. Yao, and H.Z. Huang. Early Software Quality Prediction Based on a Fuzzy Neural Network Model. in Proceedings of the Second International Conference on Innovative Computing, Information and Control. 2007. Washington DC, USA.

[27] Su, Y.-S. and C.-Y. Huang, Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. The Journal of Systems and Software, 2007. 80 p. 606-615.

[28] Shukla, R. and A.K. Misra. Estimating Software Maintenance Effort A Neural Network Approach. in In the Proceedings of the 1st India software engineering conference ISEC '08. 2008. Hyderabad, India.

[29] Mittal, H., P.K. Bhatia, and P. Goswami, Software Quality Assessment Based on Fuzzy Logic Technique. International Journal of Software Computing Applications, 2008(3): p. 105-112.

[30] Srivastava, P.R., A.P. Singh, and K.V. Vageesh, Assessment of Software Quality: A Fuzzy Multi - Criteria Approach, in Evolution of Computation and Optimization Algorithms in Software Engineering: Applications and Techniques, 2010, IGI Global USA. p. 200-219.

[31] Srivastava, P.R., et al. Software quality factor evaluation using Fuzzy multi-criteria approach. in Proceedings of the 4th Indian International Conference on Artificial Intelligence (IICAI 2009). December, 2009. Tumkur, Karnataka, India.

[32] Yang, H., Measuring Software Product Quality with ISO Standards Base on Fuzzy Logic Technique Affective Computing and Intelligent Interaction: Advances in Intelligent and Soft Computing, 2012. 137: p. 59-67.

[33] Yuen, K.K.F. and H.C.W. Lau. Evaluating Software Quality of Vendors using Fuzzy Analytic Hierarchy Process. in Proceedings of the International Multi Conference of Engineers and Computer Scientists Vol I (IMECS 2008). March, 2008. Hong Kong.

[34] Yuen, K.K.F. and H.C.W. Lau, Fuzzy group analytical hierarchy process approach for software quality assurance management: Fuzzy logarithmic least squares method. Expert Systems with Applications: An International Journal, August, 2011. 38(8): p. 10292-10302.

[35] Mago, J. and P. Kaur. Analysis of Quality of the Design of the Object Oriented Software using Fuzzy Logic. in International Conference on Recent Advances and Future Trends in Information Technology. 2012.

[36] Mittal, J.P., P. Bhatia, and H. Mittal, Software maintenance productivity assessment using fuzzy logic. ACM SIGSOFT Software Engineering Notes, Sep. 2009. 34(5).

[37] Sharma, A., P.S. Grover, and R. Kumar, Predicting Maintainability of Component-Based Systems by Using Fuzzy Logic in Contemporary Computing: Communications in Computer and Information Science, 2009. p. 581-591.

[38] Singh, Y., P.K. Bhatia, and O. Sangwan, Predicting software maintenance using fuzzy model. ACM SIGSOFT Software Engineering Notes 2009 34 (4): p. 1-6.

[39] Al-Jamimi, H.A. and M. Ahmed. Prediction of Software Maintainability Using Fuzzy Logic. in IEEE 3rd ICESS. 2012. Beijing, China.

[40] Chang, E. and T.S. Dillon, A Usability-Evaluation Metric Based on a Soft-Computing Approach. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART A: SYSTEMS AND HUMANS, March 2006. 36(2).

[41] Lin, J.-C. and K.-C. Wu. Evaluation of Software Understandability Based On Fuzzy Matrix. in in IEEE International Conference on Fuzzy Systems (FUZZ 2008). 2008.

[42] Sandhu, P.S., D.S. Salaria, and H. Singh, A Comparative Analysis of Fuzzy, Neuro-Fuzzy and Fuzzy-GA Based Approaches for Software Reusability Evaluation. World Academy of Science, Engineering and Technology, 2008.

[43] Nerurkar, N.W., A. Kumar, and P. Shrivastava, Assessment of reusability in aspect-oriented systems using fuzzy logic. ACM SIGSOFT Software Engineering Notes archive, 2010 35 (5): p. 1-5

[44] Sharmaa, R.K., D. Kumarb, and P. Kumarb, Fuzzy modeling of system behavior for risk and reliability analysis. International Journal of Systems Science, 2008 39(6): p. 563-581.

[45] Guillaume, S., Designing fuzzy inference systems from data: an interpretability-oriented review. IEEE Trans. Fuzzy Systems, 2001. 9 p. 426-443.

[46] Paiva, R.P. and A. Dourado, Interpretability and learning in neuro-fuzzy systems. Fuzzy Sets and Systems, 2004. 147 p. 17-38.

[47] Ahmed, M.A. and H.A. Al-Jamimi, Machine Learning Approaches for Predicting Software Maintainability: A Fuzzy-based Transparent Model. IET Software, 2013.