Journal Of Harmonized Research (JOHR)

Journal Of Harmonized Research in Engineering
8(2), 2020, 34-40

ISSN 2347 – 7393

Original Research Article

AN ANALYSIS ON SOFTWARE QUALITY PROPHECY USING DATA MINING AND MACHINE LEARNING PROFICIENCY

Shubha Dubey

Asst. Professor, CSE/IT Dept. Sage University, Indore, India

Abstract*:* The need of the time is highly reliable software usage in the systems. However, time-consuming costly development processes to assure reliability. One lucrative approach is to aim reliability-enhancement activities to those modules that are likely to have the most problems. If in each module software quality prophecy models can predict the number of faults at the earliest it would be expected enough effective for reliability enhancement. The main purpose of this paper is to help developers categorize defects based on existing software metrics using data mining proficiencies and by this means improve the software quality. In this paper, various software quality prophecy strategies based on Bayesian belief network, neural networks, fuzzy logic, support vector machine and case-based reasoning are juxtapose. This study gives better relative understanding about these strategies, and helps to choose an approach based on existing resources software quality prophecy using software metrics in the literature.

Keywords: Software quality prophecy, metrics, machine learning, prophecy proficiency.

Introduction: Software quality refers to software functional quality as well as software structural quality, in the reference of software engineering. Software functional quality reflects functional requirements whereas structural quality highlights non-functional requirements.

For Correspondence:
shubha.dubey07@gmail.com.
Received on: March 2020
Accepted after revision: April 2020
DOI: 10.30876/JOHR.8.2.2020.34-40

Software metrics hubs on the quality aspect of the product process and project. In this paper the key stress is on software product. The purpose of software product quality engineering is to attain the required quality of the product through the definition of quality necessities and their execution, dimension of suitable quality attributes and assessment of the resulting quality. Software quality dimension [15] is about quantifying to what amount a system or software possesses pleasing characteristics namely Reliability, Efficiency, Security, Maintainability and (adequate) Size. This can be performed through qualitative or quantitative means or a mix of both. In both cases, for each pleasing trait, there are a set of measurable attributes like Application Architecture Standards, Coding, Practices, Complexity, Documentation, Portability and Technical & Functional volumes. The survival of these attributes in a section of software or system tends to be associated and connected with this characteristic.

Software quality prediction: Predictability is a measure of how accurately a future event can be forecasted [11]. High-quality software means that it will behave in an acceptable defined manner. J. Voas [11, 22] The hurdles in prophecy can be categorized into two: hidden defects and humongous input spaces. Due to mental mistakes of programmers or designers hidden defects may occur. Complication in the programs is due to infinite input spaces. Commercial software complication is mounting hile time pressure to liberate high quality software is also on the rise. Although, more defects are likely due to enlarged software obstacle on the contrary, organizations cannot afford to let trip many bugs in the field. It is becoming exorbitant to wait for flaw till its testing phase (after development) and then fix it. It is rather possible that such a dilemma may take more time than the development of time of an exact module. Based on all former information, software quality is predicted so that faulty modules could before casted and trial should be taken to grip them in advance. In this way, software quality will be more guaranteed and hence will accomplish the needs of today's cutthroat marketplace.

AI proficiency for software quality prophecy: To effectively assure quality in software systems, prophecy of software quality has become important. There are many methods proposed so far to predict software quality, but software reliability growth model (SRGM) is one of the best-known methods [1]. Reliability and constancy of a software module are key indicators, which can be utilized to give approximate software quality. In bulk of the quality models, software product metrics are utilized to predict software quality, but in case of often changing software, this prophecy will give erroneous results. Thus, we need to integrate software system metrics as well to predict software quality. Process history can be used to predict reliability of a module when we are incorporating process metrics [18].

AI based approaches and related work
With regards to software fault prophecy; majorly research works have focused on fault modules. The complexity of software fault and maintainability is a significant part of software project development. Software size and software fault data have frequently been utilize in the advancement of models that predict software quality. A few software quality models predict the quality of software modules as also faulty or non-faulty [20], [21]. At the present time, a few of the major application areas for case-based reasoning (CBR) in the field of health sciences, multi-agent systems in addition to in Web-based planning are in trend. Research on CBR in the vicinity is mounting, but most of the systems are still prototypes and not obtainable in the market as profitable products. However, many of the systems are intended to be commercialized [22]. There is a little, but rising part of work in which machine knowledge method is utilized in the software quality (fault) prophecy task. Many researchers have utilized AI-Based approach like Case-Based Reasoning (CBR), Genetic Algorithm (GA), Neural Network (NN), etc. Khan *et al.* [7] mentioned that, when software quality was predicted, the main objective was to predict reliability and stability of the software. Becker *et al.*[8] was predicted performance of the software. Present software fault prophecy models commonly engross by means of supervised learning methods to train a software fault prophecy model. Predicting the quality of the software product through its development stage is a exigent job especially when software sizes grow. Zhong *et al.* has used unverified. Learning techniques to build a software quality estimation system [5]. Case-based reasoning has also been utilized by Kadoda *et al.* in [1].

Myrtveit *et al.* in [2] and Ganesan et. AI in [3] have also studied CBR was applied to software quality modeling of a family of complete industrial software systems and the accuracy is measured better than a corresponding multiple linear regression model inpredicting the number of design faults. Aamodtand Plazaare given the case-based reasoning cycle [10]. Rashid *et al.* emphasized on the importance of software quality prophecy and accuracy of case-based estimation model[6][9].

Learning/prophecy system: Learning or training methods can be broadly classified into three basic types [19].

- Supervised Learning: When the authentic reaction varies from target reaction then network generates a fault indication. To minimize the replace we need a supervisor or teacher, hence the name supervised learning.
- Unsupervised Learning: Unsupervised learning does not entail an administrator or teacher; it needs definite guidelines to make groups. Grouping can be done on the basis of the prototype or belongings of the object.
- Reinforcement learning: It is alike as supervised learning. All the way, in this process the teacher does not cite how nearer is the definite output to the desired output. Therefore, replace generated during reinforced learning is binary. Machine learning proficiencies have the potential to predict software quality in the early stages of software development. A number of them can be functional if previous data (Training data) is available in the knowledge base. In this research work, we have utilized case-based reasoning (CBR) as a machine learning proficiency. The bilateral learning/prophecy system can be seen in figure 1. A user can access the prophecy system using the given interface to study new troubles and get the predicted outputs (see figure1). The components of the prophecy system are as follows:

1) Interface: It is the point of interaction. **2) Estimation/ Prophecy system:** Estimating or predicting a system is typically done by gaining knowledge from the prior knowledge and provides information for future solutions to some extent.

3) Machine learning algorithm: Case-based reasoning has been utilized as a machine learning algorithm where the method of solving a new case(s) based on the solutions on similar previous cases.

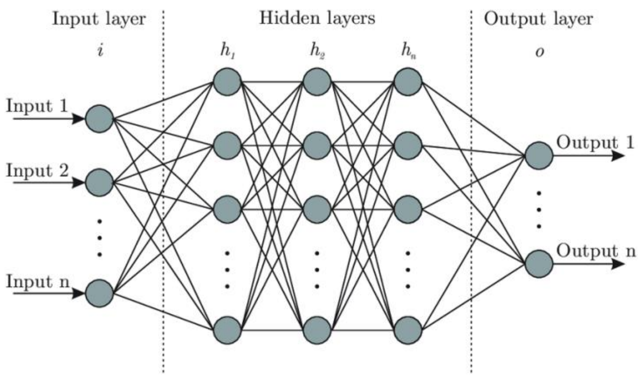
At the time when we are worried for knowing about the quality of a module to be built, we have many metrics and keys which need to be aligned in a convinced way to approximate the quality level in a clever way. Diverse artificial intelligence approaches come into play here. Following discussion spans various AI approaches and their methodology to predict software quality:

A. Bayesian Belief Network

S. Amasaki *et al.* [1] has found some projects which generate poor quality products and SRGM fails to predict quality in such cases. Regression models can be utilize to classify such perilous projects, but they suffer from a stern problem, that highly associated metrics can't be utilize concurrently and hence the prophecy results will not be reliable. S. Amasaki *et al.* [1] have proposed to use Bayesian Belief Network (BBN). BBN models the causal relationship between variables and is capable of handling probabilistic events (uncertainties). This trait enables this approach to be useful in an environment where values of all variables (metrics) are not known. In such scenarios, probabilistic approach can be functional to estimate the value of an unknown variable. BBN approach has two major parts: a directed acyclic graph (DAG) and a probability table. The DAG portrays the casual relationship between variables. The probability table is assigned conditional probabilities of each dependent variable. While executing probability calculation using BBN, values of all variables may be known or values of some variables are unknown. [1]

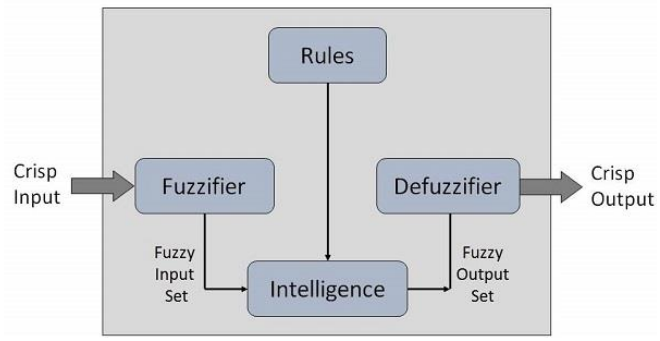
Five types of metrics have been recommended to be utilized as variables: product size, effort, detected faults, test items and residual faults [1]

B. Neural Networks: Neural network is a type of statistical computer program which categorizes gigantic and multifaceted data sets by consortium cases together in a way similar to the human brain. It is a computational method for optimizing for a preferred property based on previous learning cycles (training). Q. Wang *et al.*[13] have proposed a neural network model for software quality prophecy. As the first step of their model, a three-layer feed-forward neural network with the sigmoid function in the hidden units and the unique function in output unit is trained. Then clustering genetic algorithm is utilized to extract graspable rules from the trained network. This extracted rule set is to utilize and to detect the fault prone software modules in the later release. So, it acts as classifier which classifies a module as faulty or non-faulty [13, 20].



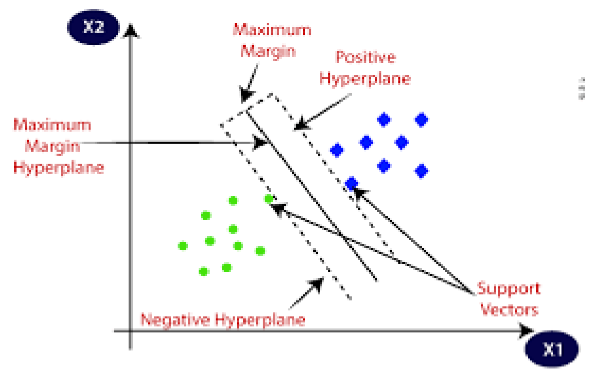
Fg. 1 Neural Networks

C. Fuzzy Logic: Mittal *et al.* (2008) use the metrics inspection rate and replace density in order to assess the software quality by using a fuzzy logic-based approach. In this approach, the software module quality can be quantified on the basis of inspection rate and replace density. A typical decision tree-based model contains a set of rules, and each rule contains one or more conditions and conclusion. In condition, typically a numerical value of a metric is compared with a threshold value. If all conditions are true then a value is assigned to the quality characteristic, which leads to the conclusion. These numerical values assigned to variables depend on the training data and can't be generalized for good prophecies. So, instead of numerical values, linguistic comparisons are made. It will lead to a fuzzy decision tree. In fuzzy decision trees, the processing of input starts by fuzzifying each of the attributes. Now, these linguistic variables take values from a set of discrete labels (e.g. bigger, smaller etc) [3, 10]. Each label has an associated membership function that sets the degree of membership based on the given numerical input. Membership functions of adjacent labels overlap, and it leads to a situation where there may be more than one membership against certain input. So, when applying this methodology on decision tree, all branches leaving out from a node are not mutually exclusive now, and all paths are observed in parallel. Only leaf nodes lead to a decision. Similarly, regression trees can also be utilize based on neural network approach [16]



Fg.2 Fuzzy Logic

D. Support Vector Machines: Many approaches of software quality prophecy use software complexity metrics as the input to quality prophecy model, because these two attributes are closely related. Generally, software modules can be classified as faulty or non-faulty modules. The relationship between software modules and their classification may be non-linear and can cause problems in quality prophecy process using traditional approaches. Support Vector Machine (SVM) is a data classification proficiency which can generalize high dimensional spaces under small training sample conditions [6].



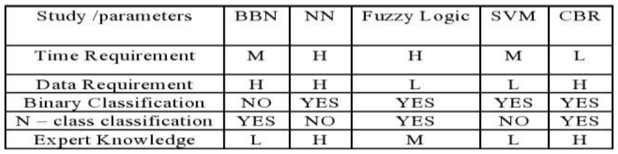
Fg.3 Support Vector Machines

E. Case-Based Reasoning: Case-Based Reasoning (CBR) is an automated reasoning approach which uses its past knowledge to solve the current problem [5]. The prior knowledge is stored in a depository called case library and it represents the prior experiences within the problem domain. Each case in the case library is stored in the form of problem description and solution pair. To crack the current problem, it is harmonized with the obtainable problems and its elucidation is forecasted based on the solutions of directly matching problems. T. M. Khoshgoftaar *et al.* [5] have proposed a methodology to predict software quality using CBR. According to their suggested approach, case base contains module (software module) description and number of faults. Module explanation is stored in form of the set of software metrics representing the module. Current module, whose quality is to be predicted, is coordinated with all case modules and its number of faults is predicted based on the number of faults in closely matched cases and the degree of matching. For the sake of quality prophecy, three major components are using similar functions, number of nearest neighbor cases are used for fault prophecy and solution algorithm. Software metrics are utilized as independent variables and numbers of faults are treated as dependent variables. Target case is compared with each case in the case library using a similarity function. Similarity functions may include Manhattan distance, Euclidean distance or Mahalanobis distance.

Comparison and discussion: Definition of quality is vague. Thus, almost every software quality prophecy model predicts quality in its own terms. Some models classify quality in one way and some in another way. They may be taking quality in terms of number of total defects, number of residual defects or just classifying a module under construction as faulty or non-faulty. BBN predicts quality in terms of residual defects. Residual defects are those defects which are present at the time of acceptance testing. An important problem with existing BBN approach as identified in [1] is that human factor in quality prophecy has not yet been catered and ignoring it at the moment may lead to loss of accuracy in prophecy. Neural network approach interprets quality in conceptual terms and classifies the module as faulty or non-faulty. Training data plays a vital role in prophecies when using this approach. This approach may work better if combined with fuzzy logic to cater the loss of accuracy at boundaries. Most of the proficiencies depend on threshold derived using sample data. These samples don't essentially represent the trends of real-life problems. So, these thresholds need not to be hard and fast. Here comes the role of fuzziness. Fuzzy decision trees are dependent on the rule base. If we succeed to form a fuzzy rule base which suitably captures the problem domain, then combining fuzzy decision trees and a fuzzy inference method, good prophecies' can be done. Furthermore, fuzzy logic approach may be improved when combined with genetic algorithms to automatically generate example data to reduce the cost and improve the accuracy [12]. SVM is a classifier and takes quality in terms of two classes: faulty and non-faulty modules. Its positive aspects is that it has the capability to work fine even in the presence of small amount of data by projecting it into high dimensional space. As far as CBR is concerned, it is based on a strong working hypothesis and that is "current module will be as faulty as previous modules with similar software metrics". This hypothesis restricts the CBR approach that the current module whose quality is to be forecasted must belong to same sphere as that of the modules in case base because it might happen that current module is an embedded system and its software metrics under consideration are behaving closely to those of modules in case base and case base contains history of modules from business domain. So, now our prophecy may be seriously affected. An ideal software quality prophecy model should be able to capture at least two significant aspects of knowledge: general knowledge of the domain and context specific knowledge of the organization. This may be achieved using a hybrid approach which is a mixture of two approaches covering these knowledge types. In this regards, S. Bouktif *et al.* [14] suggest to use genetic algorithms. Accuracy of quality prophecy and its ROI are highly correlated. Studies in [17] show that quality prophecy is highly desirable large systems from economical point of view.

Table 1 summarizes and compares various approaches with respect to different angles. In this table, one dimension shows various Al approaches to be compared and other dimension shows various parameters or attributes on basis of which these approaches have been compared.

Table I Summarized comparison of various approaches



Study /parameters	BBN	NN	Fuzzy Logic	SVM	CBR
Time Requirement	M	H	H	M	L
Data Requirement	H	H	L	L	H
Binary Classification	NO	YES	YES	YES	YES
N – class classification	YES	NO	YES	NO	YES
Expert Knowledge	L	H	M	L	H

Future work: Defect correction effort can be predicted using defect association mining [19], but data mining proficiency may be effectively applied in future to predict quality without essentially knowing before some defects. As a future work, besides this general comparison, domains of each of these approaches can be explored and experimented in detail, so that we may have a superior idea which prophecy approach to be followed in which type of projects. This work may then emphasize on combination of certain approaches or devising new approaches where all or popular one of them show ignorance.

Conclusion: Timely prophecies of software reliability and stability can be created using software quality models. The objective is to predict in advance the software quality so that resources may be allocated during software development process. According to the tendency of faults in the module. This foretell problem has been regale using diverse AI proficiencies like BBN, neural networks, fuzzy logic, SVM etc. In this paper I have compared capabilities of various approaches, and the ways they predict quality. I criticized some of the approaches and acknowledged their weak points. I recommended some prospect work after comparing these accessible proficiencies. This relative study may be helpful to spot an approach based on our desired goals and available resources.

References

[1] S. Amasaki, Y.Takagi, 0. Mizuno, and T. Kikuno, "A Bayesian belief network for assessing the likelihood of fault content," Proceedings of the 14th IEEE International Symposium on Software Reliability Engineering (ISSRE'03), 2003.

[2] D. Grosser, H. A. Sahraoui, and P. Valtchev, "Analogy-based software quality prediction," Proceedings of the 7th IEEE Workshop on Quantitative Approaches In Object-Oriented Software Engineering (QAOOSE'03), 2003.

[3] H. A. Sahraouil, M. Boukadoum, H. M. Chawiche, G. Mai, and M. Serhani, "A fuzzy logic framework to improve the performance and interpretation of rule-based quality prediction models for 00 software," Proceedings of the 26th IEEE Annual International Computer Software and Applications Conference (COMPSAC'02), 2002.

[4] J. Lee, J. Y. Kuo, and Y. Y. Fanjiang, "A note on current approaches to extending software engineering with fuzzy logic," Proceedings of the IEEE International Conference on Fuzzy Systems, 2002.

[5] T. M. Khoshgoftaar, N. Seliya, and N. Sundaresh, "An empirical study of predicting software faults with case-based reasoning," Software Quality Journal 14:85-111, Springer Science and Business Media, 2006.

[6] F. Xing, P. Guo, and M. R. Lyu, "A novel method for early software quality prediction based on support vector machine," Proceedings of the 16th IEEE International Symposium on

Software Reliability Engineering (ISSRE'05), 2005.
[7] X. Yuan, T. M. Khoshgoftaar, E. B. Allen, and K. Ganesany, "An application of fuzzy clustering to software quality prediction," Proceedings of 3 d IEEE Symposium on Application Specific Systems and Software Engineering Technology, 2000.
[8] T. S. Quah, and M. M. T. Thwin, "Application of neural networks for software quality prediction using object-oriented metrics," Proceedings of the International Conference on Software Maintenance (ICSM'03), 2003.
[9] T. M. Khoshgoftaar, E. B. Allen, X. Yuan, W. D. Jones, and J. P. Hudepoh, "Assessing uncertain predictions of software quality, "Proceedings of 6th IEEE International Software Metrics Symposium, 1999.
[10] H. A. Sahraoui, M. Boukadoum, and H. Lounis, "Building quality estimation models with fuzzy threshold values," www.iro.umontrealca/sahraouhlpapersllobjetO O02.pdf, September 18, 2006.
[11] J. Voas, "Can chaotic methods improve software quality predictions?," IEEE Software, September / October 2000.
[12] E. Baisch, and T. Liedtke, "Comparison of conventional approaches and soft-computing approaches for software quality prediction," Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 1997.
[13] Q. Wang, B. Yu, and J. Zhu, "Extract Rules from Software Quality Prediction Model Based on Neural Network," Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04), 2004.
[14] S. Bouktif, D. Azar, D. Precup, H. Sahraoui, and B. K'egl, "Improving rule set based software quality prediction: A genetic algorithm-based approach," Journal of Object Technology, vol. 3, no. 4, April 2004, Special issue: TOOLS USA 2003, pages 227-241, http:llwww.jot.fm/issues/issue 2004 04/
[15] T. M. Khoshgoftaar, E. B. Allen, R. Halstead, G. P. Trio, and R.Workshop, 1997.
[16] T. M. Khoshgoftaar, and N. Seliya, "Tree-based software quality estimation models for fault prediction," Proceedings of the Eighth IEEE Symposium on Software Metrics (METRICS'02), 2002.
[17] T. M. Khoshgoftaar, E. B. Allen, W. D. Jones, and J. P. Hudepohl, "Return on investment of software quality predictions," Proceedings of IEEE Workshop on Application-Specific Software Engineering Technology, 1998.
[18] T M. Khoshgoftaar, E. B. Allen, R. Halstead, G. P. Trio, and R. M. Flass, "Using process history to predict software quality," IEEE Computer, 1998.
[19] Q. Song, M. Shepperd, M. Cartwright, and C. Mair, "Software defect association mining and defect correction effort prediction," IEEE Transactions on Software Engineering, Vol. 32, No. 2, February 2006.
[20] N. J. Pizzi, A. R. Summers, and W. Pedrycz, "Software quality prediction using median-adjusted class labels," Proceedings of the IEEE International Joint Conference on Neural Networks, 2002.
[21] P. Guo, and M. R. Lyu, "Software quality prediction using mixture models with EM algorithm", Proceedings of 1st IEEE Asia-Pacific Conference on Quality Software, 2000.
[22] Vaseem Naiyer, Jitendra Sheetlani, Harsh Pratap Singh, "Software Quality Prediction Using Machine Learning Application", Smart Intelligent Computing and Applications, Springer, 2020.pp 319-327.