Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.754958

A NOVEL METHOD OF SOFTWARE QUALITY PREDICTION USING COMPUTATIONAL INTELLIGENCE

[1]Ms. S.M. Monisha, [2]Mr. S. Sangili, [3]Mr. S. Santhosh.
[1]Assistant Professor, Department of Computer Applications, Dr. SNS Rajalakshmi College of Arts & Science, Coimbatore.
[2]PG Student, II MCA, Department of Computer Applications, Dr. SNS Rajalakshmi College of Arts & Science, Coimbatore.
[3]PG Student, II MCA, Department of Computer Applications, Dr. SNS Rajalakshmi College of Arts & Science, Coimbatore.

Abstract: - Effective prediction of the fault-proneness plays a very important role in the analysis of software quality and balance of software cost, and it also is an important problem of software engineering. Importance of software quality is increasing leading to development of new sophisticated techniques, which can be used in constructing models for predicting quality attributes. Software quality prediction thus aims to evaluate software quality level periodically and to indicate software quality problems early. In this paper, we propose a novel technique to predict software quality by adopting Ant Colony Optimization (ACO) of software modules based on complexity metrics. Because only limited information of software complexity metrics is available in early software life cycle, ordinary software quality models cannot make good predictions generally. It consequently proposes an ACO-NM software model, whose characteristic is appropriate for early software quality predictions when only a small number of sample data are available. Therefore, proposed model was very useful in predicting software quality and classing the fault-proneness.

Keywords: Software Quality, Computational Intelligence, Prediction, ACO, Quality Assurance.

1. INTRODUCTION

The demand for software is increasing every day in various fields. Software developers put more effort to develop and test the quality of the software and verify its reliability before it is released. High-quality software modules were developed to allow others to reuse the components. In recent years, artificial intelligence has entered into various fields of life. In the field of manufacturing industries instead of human beings, many robots are engaged. These robots are equipped with brains which got trained with machine learning algorithms (ML). The relationship between artificial intelligence (AI) and software engineering (SE) is getting closer day by day, although the gap between the integration of these two fields is still large compared to the relationship between AI and other sciences [1]. A software quality model is a tool for focusing software enhancement efforts. Such models yield timely predictions on a module by-module basis, enabling one to target high-risk modules. Software metrics represent a quantitative description of program attributes and they play a critical role in predicting the quality of the resulting software. Software complexity metrics have been shown to be closely related to the distribution of faults in program modules. That is, there is a direct relationship between some complexity metrics and the number of faults later found during test, validation and operation. Consequently, investigating the relationship between the number of faults in a program and its software complexity metrics attracts attentions from many researchers.

Recent evidence indicates that most faults in software applications are found in only a few of system's components. Therefore, there has been a tremendous growth in the demand for software quality. As a consequence, issues related to predicting fault-proneness software modules are becoming increasingly crucial. Studies have shown that the majority of faults are often found in only a few software modules [2]. Such fault software modules may cause software failures, increase development and maintenance costs, and decrease customer satisfaction. Accordingly, effective prediction of fault-proneness software modules can enable software developers to focus quality assurance activities and allocate effort and resources more efficiently. This in turn can lead to a substantial improvement in software quality.

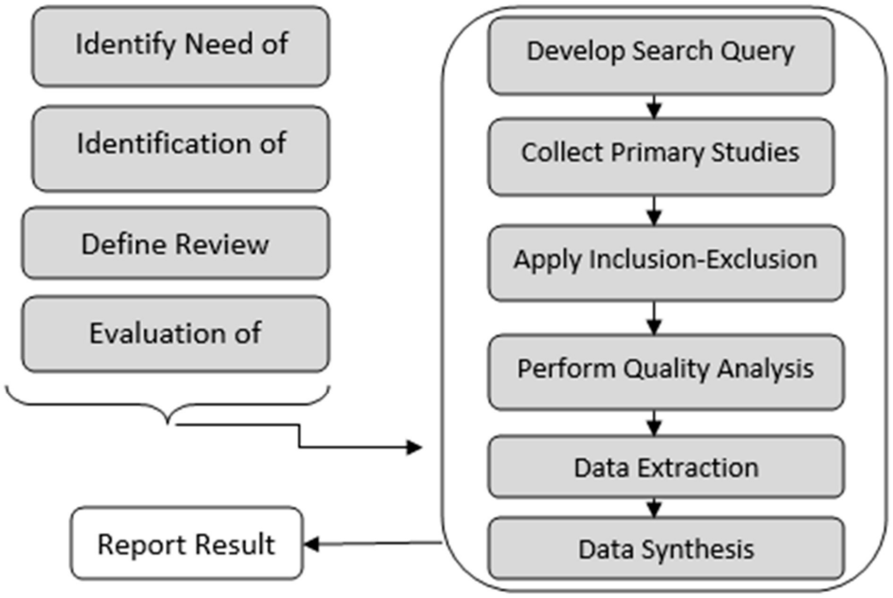

Fig.1.1: Software Quality Prediction Structure

Defect Prediction in Software is the process of determining parts of a software system that may contain defects. Application of software defect prediction models early in the software lifecycle allows practitioners to focus their testing manpower in a manner that the parts identified as "prone to defects" are tested with more rigor in comparison to other parts of the software system [3]. This leads to the reduction of manpower costs during development and also relaxes the maintenance effort. Software defect prediction models are built using two approaches: First, by using measurable properties of the software system called Software Metrics and second, by using fault data from a similar software project. Once built, the Software defect prediction model can be applied to future software projects and hence practitioners can identify defects prone parts of a software system.

Despite the emergence of various software quality prediction models and techniques, the applicability and accuracy of these models are still under research. This is due to the inherent complexity of the development process that results in the target software product, which many of the existing models have not fully taken into account. It is well known that software development is a complicated process which involves a lot of issues from technologies to management [4]. Consequently, the quality of the target software product is affected by many factors such as the characteristics of the software product, the characteristics of the development process and the operation conditions. A realistic software quality prediction model should effectively describe the influences of these factors on the quality of the target software product. To address this issue, some researchers have proposed novel approaches to software quality prediction.

Software complexity metrics can be used as input variables of quality prediction model to predict the fault number, but predicting the exact number of faults in each module is often not necessary. Several different techniques have been proposed to develop predictive software metrics for the classification of software program modules into fault-prone and non fault-prone categories. To build a predictive model, the number of changes (faults) is usually required. However, to obtain the dependent criterion variables, need to take a long time for collecting the feedback of test and validation results.

2. RELATED WORK

Modern society is fast becoming dependent on software products and systems. High reliability is one of the most important problems facing the software industry. A software quality model is a tool for focusing software enhancement efforts. Such models yield timely predictions on a module-by-module basis, enabling one to target high-risk modules [8]. Software failures can originate during different software development lifecycle phases as a result of the work done by designers, programmers, and analysts. Software testing ensures that the software is reliable and free of any such residual errors. Software quality has been measured using multiple models that use various characteristics and their relationships specifying quality requirements. These characteristics can be thought of as fundamental factors, each of which can have sub-factors). Metric based evaluation can be done for these sub-factors of software quality.

Yadav, S., & Kishan, B. (2020), focused the software product's quality; metrics and we thoroughly review the literature of existing computational intelligence/soft computing techniques and analyze the findings according to the techniques [5]. With the increase in demand for software quality prediction, several techniques have been used to predict software quality. Component-based software systems along with soft computing techniques should be used for having more reliable software. Component-based software systems along with soft computing techniques should be used for having more reliable software. This study will help the other researchers to study and understand which technique would be more helpful in making the software reliability prediction model for component-based software by combining different computational techniques.

Seyedashraf, O., Mehrabi, M., & Akhtari, A. A. (2018), knows that, Software products quality assessment is a highly complex process, given the variety of criteria to consider [6]. For a better understanding, they are organized in so-called software quality models. An important aspect of these models is their structural complexity, forming a hierarchical structure. The authors proposed a solution incorporates elements of Computational Intelligence, such as fuzzy logic, fuzzy linguistic modeling and the use of fuzzy cognitive maps (FCM). Modeling a general structure to represent quality models, extending the hierarchical taxonomies to form more complicated structures like graphs and taking into account the interrelation between criteria is the objective of this work. The application of this proposal in real-world case shows that it is an operative solution, reliable, precise and of easy interpretation for its application in the industry.

Padhy, N., Singh, R. P., & Satapathy, S. C. (2019), proposed multilevel optimization to accomplish a novel reusability prediction model. Considering coupling, cohesion and complexity as the software characteristics to signify aging proneness, six CK metrics; WMC, CBO, DIT, LCOM, NOC, and RFC are obtained from 100 WoS software [7]. The extracted CK metrics are processed for min–max normalization that alleviates data-unbalancing and hence avoids saturation during learning. The 10-fold Cross-validation followed by outlier detection is considered to enrich data quality for further feature extraction. To reduce computational overheads RSA algorithm is applied. SoftAudit tool is applied to estimate reusability of each class, while binary ULR estimates calculates (reuse proneness) threshold. Applying different classification algorithms such as LM, ANN algorithms, ELM, and evolutionary computing enriched ANN reuse-proneness prediction has been done.

3. STATEMENT OF THE PROBLEM

The software quality models play a very significant part as highly fruitful devices dedicated for the realization of the vital targets of a software quality assurance scheme. The software measures constitute the measurements of diverse features of software like the dimension, intricacy and the association between its segments [9]. Classically, the software modules are classified by a categorization model into two diverse risk-based groups, like the Faultprone (FP), and Not Fault-Prone (NFP) modules. The quality-based class membership of the modules in the training data set is habitually allocated in accordance with a pre-set threshold value of a quality factor like the number of errors, or the number of lines of code churns [10]. In the process, the software developer's scant resources may be dedicated towards the identification and rectification of errors arising only in those software modules which are highly fault-prone. The hunt for an optimal solution becomes further complex in the case of modeling with multiple software project data sets.

Software quality is regarded as the highly important factors for assessing the global competitive position of any software product. Due to lack of insufficient tools to evaluate and predict software quality one of the major challenges in software engineering field. To assure quality, and to assess the reliability of software products, many software quality prediction models have been proposed in the past decades. In this proposed method we have utilized a hybrid method for quality prediction. The prediction is done with the help of the Advanced Computational Intelligence Neural network which is incorporated with Hybrid Cuckoo search (HCS) optimization algorithm for better prediction accuracy.

4. OBJECTIVE THE RESEARCH WORK

There are models to partially generate the code or estimate the cost, effort, and quality of the required software. Many researchers have proven the ability of machine learning techniques to provide information and take measures that will speed up the process of developing software and sustain its efficiency [11]. Not only at the level of processes, but at the level of the project as a whole, including the calculation of efforts, resources and financial issues as well as the time required for release. The main objective of this work is to circumvent the model generalisability problem. Here, propose a new approach that substitutes traditional ways of building prediction models which use historical data and machine learning techniques [12].

Software quality can be described from a practical perspective by a set of attributes. Each of these attributes determines one aspect of software quality. These attributes are reliability, functionality, efficiency, maintainability, portability, and usability. These attributes can be measured based on three different types of metrics, process metrics, project metrics, and product metrics. These metrics represent the state.

The main contributions of this work are summarized below: -

- The developer needs quality assurance can only be done after obtaining experience that helps in predicting possible error spots or defects.
- Required a model to predict the value of several quality attributes (reliability, performance, maintainability, etc.) of target software product.
- Need a Model for early prediction of software quality becomes feasible and the management can have knowledge of the quality of target software product as early as possible, which helps to identify design errors and avoid expensive rework.

5. PROPOSED METHODOLOGY

The quality of a software product can be defined as the measure of performance of a system on which the software is implemented in terms of execution time, memory capacity utilized and probability of errors, etc. In addition to this, the amount of effort contributed by the software developer also represents a key factor while assessing the quality of a software product [13]. The quality of a software product can be considered to be internal as well as external. The external quality also depends upon its internal quality. In order to assess the external quality of a software product, quality models can be devised that represent a function of the internal quality attributes. In order to achieve this, first of all the internal attributes must be identified and then the relationship existing between the internal and external quality attributes must be identified.

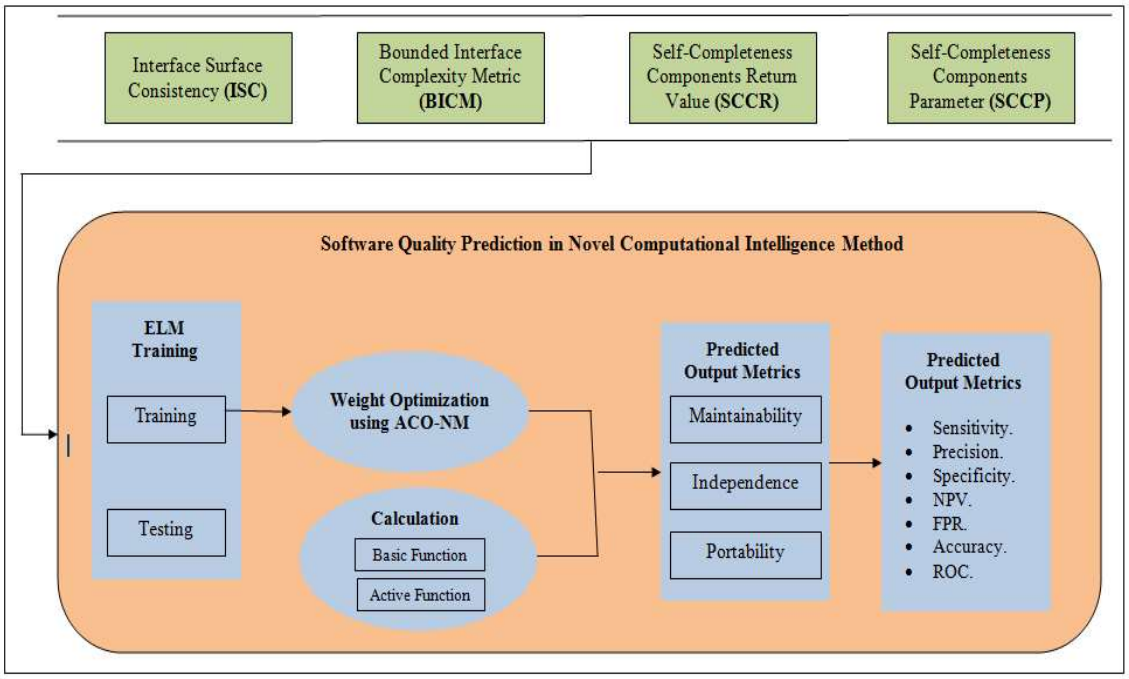
Interface Surface Consistency (ISC)
Bounded Interface Complexity Metric (BICM)
Self-Completeness Components Return Value (SCCR)
Self-Completeness Components Parameter (SCCP)
Software Quality Prediction in Novel Computational Intelligence Method
ELM Training
Training
Testing
Weight Optimization using ACO-NM
Calculation
Basic Function
Active Function
Predicted Output Metrics
Maintainability
Independence
Portability
Predicted Output Metrics
Sensitivity.
Precision.
Specificity.
NPV.
FPR.
Accuracy.
ROC.

Fig. 5.1: Workflow of Proposed Methodology

The proposed work introduces the CI-NN with the ACO–NM algorithm for SQP to transform the following software constraints into objective functions. The proposed CI-NN model is designed on the basis of the single-layer feed forward network (SLFN). Therefore, a non-linear differentiable transfer function is used in the CI-NN approach for the activation function. Then, introduced a new efficient optimization method called ACO–NM to optimize the software quality metrics components like portability, independency, and maintainability. This optimization algorithm minimizes the network error by adjusting the weight value of the NN.

In this proposed work, four input variables of interface surface consistency (ISC), bounded interface complexity metric (BICM), self-completeness components return value (SCCR) and self completeness components parameter (SCCP) are taken in the CI-NN classifier for both training and testing process. Then the hybrid ACO–NM algorithm is used for updating the optimal weight value. Finally, three expected output variables are taken in to account for better quality prediction measures are maintainability, independence, and portability. After that, the performance of the proposed work is analyzed in terms of sensitivity, precision, specificity, NPV, FPR, FDR, Accuracy, MCC and ROC. ACO is a type of meta-heuristic optimization algorithm inspired by the behavior of the real ants in which optimization problem are considered in terms of combinatorial. Whereas Nelder–Mead is a simple direct search method used to solve unconstrained local nonlinear optimization problems. In ACO, ants deposit pheromones based on the response of other ants to minimize objective function, but in NM a simplex moves from an unconstrained search space to minimize objective function. Therefore, the CI-NN follows the ACO–NM algorithm for the weight optimization space in SLFN. The hybrid combination of ACO with NM is used to perform global optimization considering non-functional quality attributes such as maintainability, independence and portability. In hybrid ACO–NM algorithm, ACO performs exploration and NM algorithm performs exploration to quickly achieve global optimum, once the ACO algorithm finds a promising region.

6. PERFORMANCE METRICS

The executions of the three methods were assessed using the predictive classification table, known as Confusion Matrix by calculating the standard metrics of accuracy, precision, sensitivity, specificity and F-measure [14].

To distinguish the instances of different classes the confusion matrix acts as a valuable tool for evaluating the algorithms. It reveals the amount of correct and wrong predictions prepared by the model compared with the actual categorizations in the dataset. True Positive (TP) and True Negative (TN) are helpful to identify, when the algorithm is generating the actual data. The parameter for the evaluation measure such as False Positive (FP) and False Negative (FN) are used to know, when the classifier is producing the faulty information [15].

- **Accuracy:** It is the proportion between the quantity of right predictions and complete number of predications.
- *acc=TP+TNTP+TN+FP+FN***acc=TP+TNTP+TN+FP+FN**
- **Precision:** It is the proportion between the quantity of right positives and the quantity of true positives in addition to the quantity of false positives.
- *(p)=TPTP+FP***Precision(p)=TPTP+FP**
- **Recall:** It is the proportion between the quantity of right positives and the quantity of true positives in addition to the quantity of false negatives.
- *recall=TPTP+FN***recall=TPTP+FN**
- **F-score:** It is known as the consonant mean of precision and review.
- *acc*=**112(1***p*+1*r*)=2*prp*+*r***acc=112(1p+1r)=2prp+r**

7. CONCLUSION

The software development process imposes major impacts on the quality of software at every development stage; therefore, a common goal of each software development phase concerns how to improve software quality. Software quality prediction thus aims to evaluate software quality level periodically and to indicate software quality problems early. Quality assurance can only be done after obtaining experience that helps in predicting possible error spots or defects. We find that artificial intelligence algorithms in general and machine learning, in particular, played their required role after quickly gaining experience through training on previously defective software. In this paper, the computational intelligence methodology is suggested to improve the performance of the prediction of software quality.

REFERENCES

[1] Jin, C., Jin, S. W., Ye, J. M., & Zhang, Q. G. (2009, December). Quality prediction model of object-oriented software system using computational intelligence. In 2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS) (Vol. 2, pp. 120-123). IEEE.

[2] Xing, F., Guo, P., & Lyu, M. R. (2005, November). A novel method for early software quality prediction based on support vector machine. In 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05) (pp. 10-pp). IEEE.

[3] Bhuyan, M. K., Mohapatra, D. P., & Sethi, S. (2016). Software Reliability Assessment using Neural Networks of Computational Intelligence Based on Software Failure Data. Baltic Journal of Modern Computing, 4(4).

[4] Khoshgoftaar, T. M. (Ed.). (2012). Software engineering with computational intelligence (Vol. 731). Springer Science & Business Media.

[5] Yadav, S., & Kishan, B. (2020). Component-based software system using computational intelligence technique for reliability prediction. International Journal, 9(3).

[6] Seyedashraf, O., Mehrabi, M., & Akhtari, A. A. (2018). Novel approach for dam break flow modeling using computational intelligence. Journal of Hydrology, 559, 1028-1038.

[7] Padhy, N., Singh, R. P., & Satapathy, S. C. (2019). Enhanced evolutionary computing based artificial intelligence model for web-solutions software reusability estimation. Cluster Computing, 22(Suppl 4), 9787-9804.

[8] Wieczorowski, M., Kucharski, D., Sniatala, P., Pawlus, P., Krolczyk, G., & Gapinski, B. (2023). A novel approach to using artificial intelligence in coordinate metrology including nano scale. Measurement, 217, 113051.

[9] Gheraibia, Y., Kabir, S., Aslansefat, K., Sorokos, I., & Papadopoulos, Y. (2019). Safety+ AI: A novel approach to update safety models using artificial intelligence. IEEE Access, 7, 135855-135869.

[10] Pachouly, J., Ahirrao, S., Kotecha, K., Selvachandran, G., & Abraham, A. (2022). A systematic literature review on software defect prediction using artificial intelligence: Datasets, Data Validation Methods, Approaches, and Tools. Engineering Applications of Artificial Intelligence, 111, 104773.

[11] Khaliq, Z., Farooq, S. U., & Khan, D. A. (2022). Artificial intelligence in software testing: Impact, problems, challenges and prospect. arXiv preprint arXiv:2201.05371.

[12] Hourani, H., Hammad, A., & Lafi, M. (2019, April). The impact of artificial intelligence on software testing. In 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT) (pp. 565-570). IEEE.

[13] Miholca, D. L., Czibula, G., & Czibula, I. G. (2018). A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. Information Sciences, 441, 152-170.

[14] Li, Y., Guo, J. E., Sun, S., Li, J., Wang, S., & Zhang, C. (2022). Air quality forecasting with artificial intelligence techniques: A scientometric and content analysis. Environmental Modelling & Software, 149, 105329.

[15] Amalfitano, D., Faralli, S., Hauck, J. C. R., Matalonga, S., & Distante, D. (2023). Artificial intelligence applied to software testing: A tertiary study. ACM Computing Surveys, 56(3), 1-38.