ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)

Vol 9, Issue 8, August 2022

A Software Quality Prediction Model for Aspect Oriented System using Neuro-Fuzzy Approach

[1] Ritu, [2] O. P. Sangwan

[1] [2] Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, India.
Corresponding Author Email: [1] Rituchopra84@gmail.com, [2] sangwan_op@yahoo.co.in

Abstract*— Nowadays, the usage of software has increased exponentially in various fields like education systems, industries, health systems, and many others. Various software architectures are already available in the market e.g. modular oriented, component-based, object-oriented, aspect-oriented, etc. Aspect-Oriented (AO) system software has gained much attention due to its superior features to the aforementioned systems. However, AO systems face the challenges of being complex and hard testing environment, a quality assessment of these systems is necessary. In this paper, a software quality estimation model for aspect oriented system using neuro-fuzzy approach has been developed. For which, a detailed study on aspect oriented systems has been accomplished in terms of various attributes affecting the quality of these software. In this paper, a framework of software quality prediction model has been designed using the adaptive neuro-fuzzy inference engine (ANFIS) approach. Data of 200 software pieces have been collected in this study where 150 software data is used to train the ANFIS model whereas 50 software data is used for testing purposes. The quality estimated by the proposed ANFIS model is compared with the actual quality of these software data and quantitative analysis is performed in terms of error measures. Finally, it was found that the proposed ANFIS model worked better in terms of MSE, MRE, MARE, MBRE, and MIBRE error measures.

***Index Terms* - ANFIS, Aspect Oriented (AO), Object Oriented Programming (OOP), Neuro-Fuzzy, Software Quality.**

I. INTRODUCTION

The survival is not easy without software. Recent developments in cheaper and affordable internet connectivity and online purchasing result into an exponential increase in the usage of computer software [1]. A number of software is available in the market for online education. A customer chooses particular software depending upon its quality which can't be numerically calculated so the purchaser looks for other customer reviews or expert's opinion and then selects a particular platform. From machine-level languages to procedural programming, OOP, Component-Based Systems and Aspect Oriented System, software development has come a long way. Even with the software approaches that are used in an industry, there is a big gap between understanding the system goals and putting them into practice. Many scholars have concluded that the best way to design manageable systems is to identify and isolate the system concerns, based on a fundamental principle of software engineering and thus the need for aspect oriented systems arises. Also, the growing need for excellent-quality software has necessitated the use of quality metrics to assess the program as well which is missing in the current literature.

A. Why Aspect Oriented Systems?

A number of software development architectures are available in the market [3] for example module oriented, component based and object oriented architectures. Object-Oriented programming (OOP) architecture [2], [4], [5] is the most widely used programming paradigm today. The goal of OOP is to arrange an application's data and accompanying procedures into logical entities and hence it promotes the reuse of software. However, when a sophisticated program comprises the following: cross-cutting concerns (e.g. logging, synchronization, etc.) and scattering of code, it might not possible to demonstrate a clear and fine programming structure in OOP.

AO provides a solution to these long-standing design and maintenance issues which have plagued software developers. In 1997, Kiczales et al. [6] proposed AOP programming paradigms with the goal of improving software quality through improved modularization and separation of concerns (SoC). Use of aspect languages is the primary focus of AO systems which offers many advantages. AO is mostly developed using AspectJ language (a java language extension) along with AspectC and AspectC++ which are extensions of C and C++ languages respectively [7]. AO leverages the concept of Aspects and demonstrates a method to extract cross-cutting concerns from various modules and arrange them into a single repository which results into no cross-cutting issues [11], [12]. Furthermore, Synchronization, consistency checking, protocol management, and other features are also offered by the AO systems.

B. Need of automatic quality assessment of aspect oriented systems

The concerns regarding software design are highly modularized to suit all of a system's non-functional requirements. Next, if the integration is not done carefully in software, any new addition to the current code may aggravate the problem even more. Because the target application's behavior will be altered, software quality factors may be impacted. Upgrading, maintaining, and modification capability of software can be simplified using the AO paradigm. However, if the AO paradigm is used incorrectly, the software may not achieve the intended degree of quality.

In addition, because of many different process paradigms and product standards, quality evaluation is necessary.

Moreover, adopting a robust methodology is crucial to ensure higher quality software. Software quality test can be divided into two categories: Manual test and Automatic test. Manual test corresponds to the manual inspection by the domain expert which is time consuming process and resource oriented as well. There may be need to invest a large amount of money on the manual test to hire a prominent expert. Whereas, automatic quality test refers to the automatic quality checking by just feeding some attributes of the software in need and thus it save both time and money of a customer and even of an organization.

However, a concrete mathematical model is impossible to be derived for the software quality as there is no well-defined phenomenon of the parameters affecting the software quality. Although, various software quality models exist in literature [13], [14], [15], the accuracy and application of these model is not in consensus because of a complex structure of various software developing platform stages. Instead, we may have a data of different software already used by the customers. Soft- computing techniques e.g. fuzzy logic [8], neural network [9], neuro-fuzzy, genetic algorithm, support vector machine etc. provide a way to learn a black box model by employing the data collected. After learning the model, these soft computing architectures can be used to automatic software quality estimation where the only requirement is to feed the necessary user inputs.

II. RELATED WORKS

There are various studies performed on the aspect oriented systems to predict various characteristics of software. An extensive literature review has been conducted on software metrics prediction in order to develop the proposed model. Authors in [16] studied an evolution of the aspect oriented programming which is followed by the software reusability analysis using fuzzy logic technique by taking four metrics of a software namely SOC, size, coupling and cohesion.

In [17], scientists investigated a study on the software quality assessment of the aspect oriented systems which employs a feed- forward neural network. However, a detailed analysis of software quality prediction is missing in this research. In paper [18], authors discussed a maintainability analysis of aspect-oriented system by considering the changeability as a measure of maintainability where authors measures the changeability at the code level in a particular module and then impact of change is analysed. The study is conducted on twenty modules of three software. Authors in [19] presented software quality attributes as internal and external matrices where internal matrices contains the sub-characteristics like SOC, coupling, cohesion etc. which directly influences the external attributes of the software quality e.g. maintainability, reusability and understandability. These three external attributes are modelled using fuzzy logic approach using three internal parameters and a software namely ’AJhotdraw’ has been taken as case study. However, no conclusion about software quality influenced by these internal and external attributes is made.

A four parameter based software quality prediction scheme is developed in [20] by employing both neural network and fuzzy logic techniques. However, it lacks a thorough comparative analysis and a quantitative analysis among presented approaches. The paper [21] discussed about maintainability analysis for aspect oriented systems using neuro-fuzzy approach with four input parameters namely CDA, CAE, CFA, WOM and the results are compared with the fuzzy logic technique. An overview of soft computing approaches has been derived in [22] to estimate the software reliability model which includes chaos theory, Bayesian function, genetic algorithm and neural network. In [23], researchers discussed about software reliability estimation for aspect oriented software using fuzzy logic technique along with genetic algorithm to tune various fuzzy logic parameters. Authors employed four internal attributes e.g. coupling, portability, interface complexity, understandability, and customizability and estimate the software reliability value.

The authors [24] provided a novel approach to solving the problem of operational profile uncertainty by evaluating dependability using Bayesian two-step inference. The technique is reliant on the availability of information regarding the entry space splitting. The author [25] provides a model for improving software dependability based on the new Gaussian distribution, which allows users to examine software process faults and reduce the uncertainty produced by human subjective elements in the software. Other soft computing techniques are also presented in the literature as shown in Table 1 which shows the analysis of various software attributes. From the above analysis, it can be seen that there are a few studies focus on the software quality prediction directly.

Table 1. Factor affecting software quality

Sr. No.	Author → Factors ↓	Kumar et al.[14]	Jagat et al. [32]	Singh et al. [33]	Puneet et al. [19]	Juneja et al. [34]	Pankaj kumar [35]	Chauhan et al. [36]	Amin et al. [37]	Sharma et al. [38]	Sangwan et al. [29]	Singh et al. [16]	Srivastava et al. [39]	Kassab et al. [40]	Total
1	SOC				*							*	*	*	4
2	Cohesion				*							*	*	*	4
3	Coupling				*	*				*		*	*	*	6
4	Size				*			*		*			*		4
5	Maintainability	*	*	*	*		*	*	*				*		8
6	Efficiency	*	*					*							3
7	Usability	*	*					*							3
8	Functionality	*	*				*	*							4
9	Modularity	*		*			*	*							4
10	Code-reducibility	*					*								2
11	Interface Complexity	*			*		*	*		*					5
12	Reusability	*	*				*	*							4
13	Reliability		*				*	*							3
14	Portability		*			*		*	*	*					5
15	Suitability	*	*												2
16	Accuracy	*	*				*	*							4
17	Interoperability	*	*				*	*							4
18	Security		*				*								2
19	Time Behavior		*				*	*							3
20	Resource Behavior	*	*				*								3
21	Scalability		*												1
22	Analyzability	*	*	*			*	*							5
23	Changeability	*	*				*	*							4
24	Testability	*	*	*			*	*							5
25	Stability	*	*	*			*		*						5
26	Track ability		*												1
27	Replace ability	*	*				*								3
28	Adaptability	*	*	*			*								4
29	Install ability	*	*				*								3
30	Understandability	*	*	*	*	*	*	*	*						8

Based on the existing research, five most important factors have been identified named coupling, maintainability, interface complexity, portability and understandability. The short description of these factors is given as:

- **Coupling:** Coupling means how closely the packages (aspects) are connected with each-other. That means it express the strength between aspects [14][38].
- **Maintainability:** It is defined as the degree to which software can be easily maintained or modified. Modifications can have debugging, up-gradation etc [32].
- **Interface Complexity:** It is defined as the degree to which software interface i.e. user's interactions are complex [35].
- **Portability:** It is defined as the ability of transferring software from one environment to other. The environment can be the combinations of hardware and software platforms [37].
- **Understandability:** It is referred to the concept by which software can be presented so that a software engineer could be able to understand software code [34].

Motivated by the above discussion, this paper infers a study on the usage of aspect oriented systems and their quality estimation at the software stage. As discussed previously, researchers have developed a number of quality models to assess the external software properties such as reusability, maintainability, understandability, reliability, testability, and efficiency individually. Although, these metrics directly affect the software quality, an individual assessment may not be enough to access the quality of software in final or deliverable stage.

Hence, this study bridges the gap in the literature and an attempt has been made to describe an effective relationship directly between software quality and these external software characteristics. To drive this relationship, a neuro-fuzzy technique via an adaptive neuro-fuzzy inference engine has been proposed which require input parameters that can affect the software quality. A through study has been performed to select these parameters of the neuro-fuzzy approach. After a close review of various metrics that can affect the software quality of an aspect oriented system, we came on conclusion of selecting five input attributes. A neuro-fuzzy architecture has been developed by employing a dataset of 150 software where after training the architecture, the quality of 50 software is estimated using this architecture. Moreover, the performance of the proposed model has been analysed quantitatively in terms of performance indexes derived by some error measures where we see that the quality predicted by the neuro-fuzzy architecture closely matches with the actual quality of these software and hence we get very low values of error measures which show the effectiveness of the proposed approach. A few studies are shown in table 2.

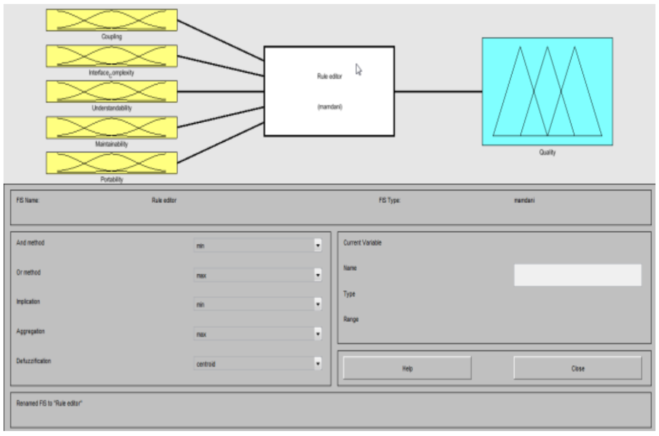
Table 2. Various studies of aspect oriented systems

Sr. No.	Objective	Method	Comment
1	Software quality assessment [26]	Neuro-fuzzy Approach	An enhanced NN-based technique is given using the Hybrid Cuckoo search optimization algorithm (HCS). To improve accuracy and speed up the learning process, the weightsare learned using the HCS algorithm. However, it lacksa thorough examination of the framework's impact on software quality.
2	Prediction of software quality [27]	Neuro-fuzzy Approach	An adaptive neuro-fuzzy technique for a web-based software quality prediction scheme is provided, with six soft- ware metrics as input. In the AKTC warehouse, a total of 682 data points were generated for specific software. A back-propagation approach yielded a testing error of 0.047 in terms of MSE for 171 data pairs.
3	Assessment of software reusability [28]	Fuzzy logic	A bug detecting automated system is constructed based on fuzzy logic technique that can discriminate between a bus and a customer request for software enhancement. There may be a big number of consumers for a given product, making manual error detection difficult, and therefore this automated solution can assist in auto-detecting the software fault.
4	Reusability prediction in software [29]	Neuro-fuzzy Approach	A thorough analysis of software internal attributes affecting the software reliability and then reliability prediction is accomplished using adaptive neuro-fuzzy inference engine (ANFIS) model.
5	Maintainability analysis of software [30]	ANFIS model	Software Maintainability analysis is performed using nine internal attributes of software. A framework of maintainability is proposed on which a model can be built to estimate the software maintainability.
6	Software defect prediction [31]	Bayesian technique	On various data sets, the model considers the connection between software metrics and faults. The number of developers and the quality of the source code are two aspects to consider.

III. VARIOUS SOFTWARE CHARACTERISTICS

The software characteristics can be divided into two categories namely internal and external characteristics where internal characteristics refer to the sub-characteristics that can influence directly a particular external characteristic/attribute. These external characteristics directly impact the software quality and can be considered as most influencing parameters to build a software quality model. Various Internal Characteristics are cohesion, coupling, size, SOC, etc. External characteristics include: portability, maintainability and understandability. Main attribute is software quality.

ISSN (Online) 2456 -1304
International Journal of Science, Engineering and Management (IJSEM)
Vol 9, Issue 8, August 2022



FIS Name: Rule editor
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

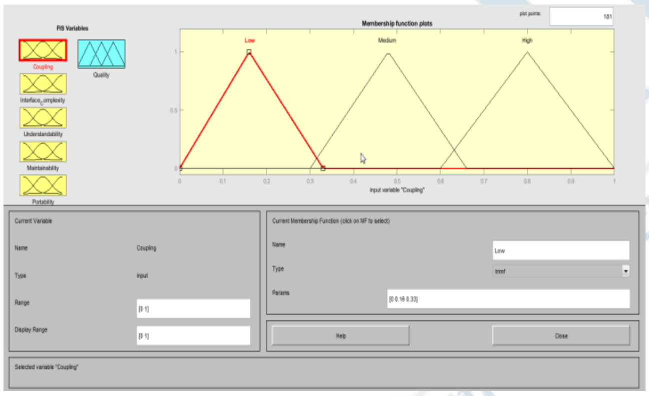
Current Variable
Name
Type
Range

Renamed FIS to "Rule editor"

Fig. 1. ANFIS Model in MATLAB with five input and one output

IV. CASE STUDY

A case study has been conducted on 200 software data whose characteristics are collected by existing research and experiences. These characteristics are 'coupling', 'interface complexity', 'understandability', 'maintainability', 'portability' and 'quality of software'. Out of 200 software data, we trained the ANFIS model as discussed in the previous section with 150 dataset of software. After training the ANFIS, we obtained optimal parameters of ANFIS model which can be used to predict the quality of the remaining 50 software data.



Current Variable
Name: Coupling
Type: input
Range: [0 1]
Display Range: [0 1]

Current Membership Function (click on MF to select)
Name: Low
Type: trimf
Params: [0 0.16 0.33]

Selected variable "Coupling"

Fig. 2: Membership function of each input parameter

A. Simulation results

First, we built an ANFIS architecture in MATLAB using five inputs namely 'coupling', 'interface complexity', 'understandability', 'maintainability', 'portability' and one 'output' as shown in Fig. 1 using Mamdani fuzzy model and feed-forward neural network. Each of the input parameter is divided into three fuzzy regions namely 'low', 'medium' and 'large' as shown in Fig. 2 whereas output i.e. software quality is divided into five regions to include more flexibility while estimating the quality. These five regions are 'very low', 'low', 'medium', 'large' and 'very large'.

Next, we train ANFIS model using 150 data pairs using grid partition algorithm in MATLAB toolbox. We estimated the quality of these 150 data pairs to demonstrate the prediction accuracy of the ANFIS model which shows how well the ANFIS model is trained. The training results are shown in Fig. 3 which shows that ANFIS model has been trained well as predicted quality and actual quality almost re-overlapped to each other.

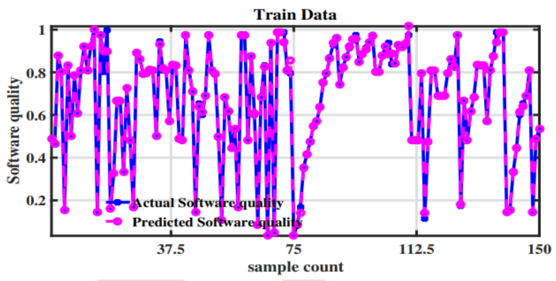


Fig. 3. Training results of presented approach

The testing results are shown in Fig. 5 which indicates good prediction accuracy of software quality by the proposed ANFIS model.

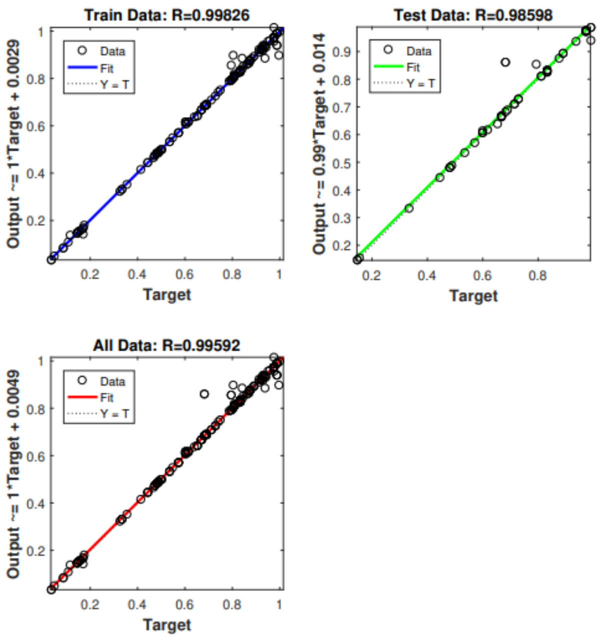


Fig. 4. Regression plot

Next, regression plot and surface plots are also shown in Fig. 4 and 6 respectively.

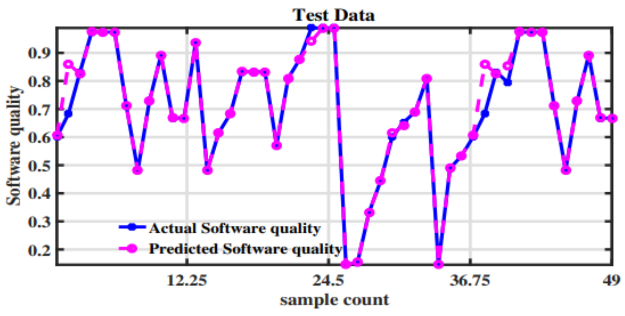


Fig. 5. Testing results of proposed model

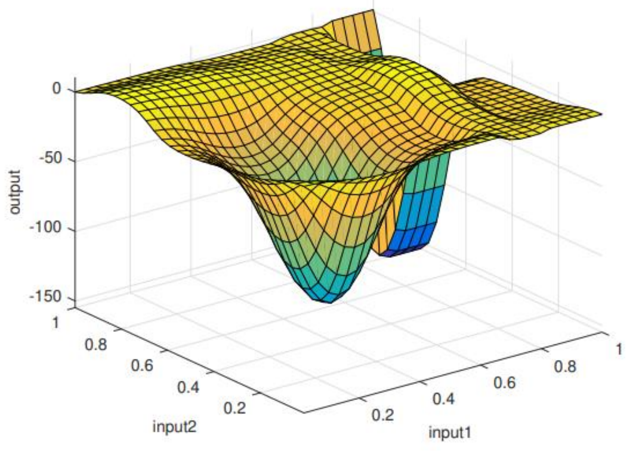


Fig. 6. Surface plot

Rule base for an input pair is shown in Fig. 7 which shows fired rules for the input data pair.

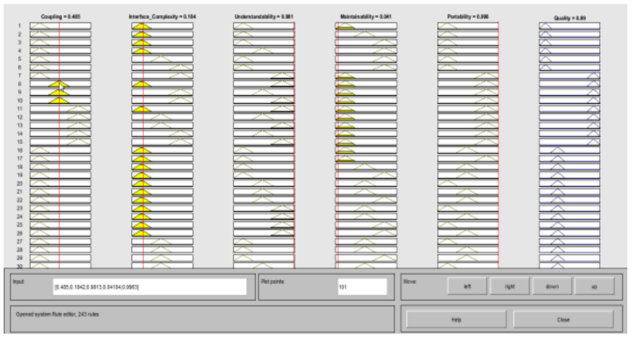


Fig. 7. Rule base for an input data pair

B. Quantitative analysis

Five performance indexes are employed to test the prediction accuracy of the proposed ANFIS model. These performance indexes are: MSE, MRE, MARE, MBRE, and MIBRE.

Table 3. Quantitative analysis of prediction accuracy of ANFIS in terms of performance measures

Sr. No.	Performance Parameter	Training	Testing
1	MSE	0.0002	0.0014
2	MARE	0.0082	0.0155
3	MRE	-0.0010	-0.0114
4	MBRE	0.0086	0.0155
5	MIBRE	0.0078	0.0132

Table 3 shows the obtained results of these performance indexes which show lower values of these measures. Lower values of these measures indicate a good prediction accuracy of the model. Hence, it is concluded that the proposed ANFIS model can correctly predicts the quality of unknown software based on just five attributes.

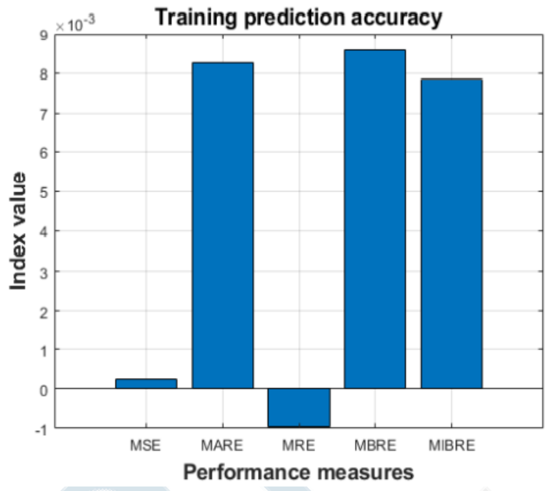


Fig. 8. Performance measures values in the training phase

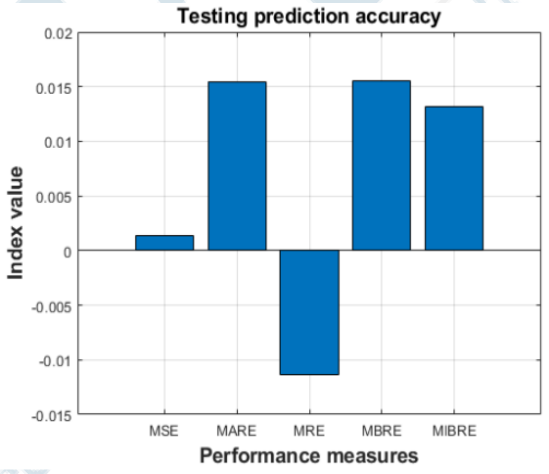


Fig. 9. Performance measures values in the testing phase

V. CONCLUSION

Aspect oriented (AO) systems employs Aspect language which is an extension of conventional programming language of object oriented (OO) system, hence, AO system inherits the properties of OO systems and attributes like SoC that indirectly affect the software quality. This paper proposes an ANFIS model to predict the software quality of the Aspect Oriented Systems based on five inputs named coupling, maintainability, interface complexity, portability, and understandability. A total of 200 software data pairs have been collected where 150 data pairs are used for ANFIS training. After ANFIS training, optimal weights parameters are obtained which are used to predict the quality of the remaining 50 software. Furthermore, a quantitative analysis of the predicted quality by the proposed ANFIS has been performed by employing the error measures where the error denotes the differences between the predicted software quality and actual software quality. Finally, lower values of error measures are obtained which shows the potential of the proposed approach. This approach may be implemented in real-time scenario that can save money and time of the respective customers.

REFERENCES

[1] P. Kumar, S. K. Singh, and S. D. Choudhary, “Reliability Prediction Analysis of Aspect-Oriented Application using Soft Computing Techniques,” Materials Today: Proceedings, vol. 45, pp. 2660-2665, 2021.

[2] F. Alaswad, and E. Poovammal, “Software Quality Prediction using Machine Learning,” Materials Today: Proceedings, vol. 62, no. 7, pp. 4714-4720, 2022.

[3] Y. R. Kirschner, “Model-Driven Reverse Engineering of Technology-Induced Architecture for Quality Prediction,” European Conference on Software Architecture, 2021.

[4] W. B. Rebecca, B. Wilkerson, and L. Wiener, “Designing object-oriented software,” 1990.

[5] P. Wegner, “Concepts and paradigms of object-oriented programming,” ACM Sigplan Oops Messenger, vol. 1, no.1, pp. 7-87, 1990.

[6] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. M. Loingtier, J. Irwin, “Aspect-oriented programming, European Conference on Object-Oriented Programming, pp. 220-242, 1997.

[7] R. Kumar, Dilip and M. Rai, A Comparative Study of AOP Approaches: AspectJ, Spring AOP, JBoss AOP,” World Congress on Engineering and Computer Science, pp. 22-24, 2019.

[8] D. Silva, and W. Clarence, “Intelligent Control: Fuzzy Logic Applications,” CRC Press, 2018.

[9] M. Vanmali, M. Last, and A. Kandel. ”Using a Neural Network in The Software Testing Process,” International Journal of Intelligent Systems, vol. 17, no.1, pp. 45-62, 2002.

[10] Y. Kamei, A. Monden, and K. Matsumoto, “Empirical Evaluation of SVM-Based Software Reliability Model,” ACM-IEEE Int’l Symposium on Empirical Software Engineering, vol. 2, 2006.

[11] M. I. Ghareb, and G. Allen, “Quality Metrics measurement for Hybrid Systems (Aspect),” Technium, vol. 3, no. 3, pp. 820-99, 2021.

[12] S. Clarke, and E. Baniassad, “Aspect-Oriented Analysis and Design,” Addison-Wesley Professional, 2005.

[13] P. Kumar, “Aspect-Oriented Software Quality Model: The AOSQ Model,” Advanced Computing: An International Journal, vol. 3, no. 2, pp. 105-118, 2012.

[14] A. Kumar, P. S. Grover, and R. Kumar, “A Quantitative Evaluation of Aspect-Oriented Software Quality Model”, ACM SIGSOFT, vol. 34, no.5, 2009.

[15] Suman, and M. Wadhwa, “A comparative Study of Software Quality Models,” International Journal of Computer Science and Information Technologies, vol. 5, no. 4, pp. 5634-5638, 2014.

[16] P. K. Singh, O. P. Sangwan, A. P. Singh, and A. Pratap, “A Framework for Assessing The Software Reusability using Fuzzy Logic Approach for Aspect Oriented Software,” International Journal of Information Technology and Computer Science, vol. 7, no. 2, pp. 12-20, 2015.

[17] P. Kumar, S. Dixit, and S. K. Singh, “Performance of Aspect-Oriented Software Quality Modelling using Artificial Neural Network Technique” International Journal of Computer Application, vol. 182, no. 36 pp. 6-10, 2019.

[18] A. Kumar, R. Kumar, and P. S. Grover, “An Evaluation of Maintainability of Aspect-Oriented systems: A Practical Approach,” International Journal of Computer Science and Security, vol. 1, no. 2 pp. 1-9, 2007.

[19] P. J. Kaur, and S. Kaushal. ”A Fuzzy Approach for Estimating Quality of Aspect Oriented Systems,” International Journal of Parallel Programming, vol. 48, no. 5, pp. 850-869, 2020.

[20] A. K. Gupta, R. Masood, “Analysis of Object Oriented System Quality Model using Soft Computing Techniques,” International Journal for Research & Development in Technology, vol. 7, no. 5, pp. 587-597, 2017.

[21] M. Hossein, and S. Zahedian, “Aspect-Oriented Software Maintainability Assessment Using Adaptive Neuro Fuzzy Inference System (ANFIS),” Journal of Mathematics and Computer Science, vol. 12, no. 3, pp. 243-252, 2014.

[22] P. Dhavakumar, S. Shankar, P. M. Vikram “Soft Computing Techniques for Enhancing Software Reliability,” International Journal of Latest Trends in Engineering and Technology Special Issue, pp. 133-140, 2018.

[23] https://dias.ac.in/wp-content/uploads/2020/03/08-15software quality-prediction-in.pdf

[24] R. Pietrantuono, P. Popov, and S. Russo, “Reliability Assessment of Service-based Software under Operational Profile Uncertainty,” Reliability Engineering & System Safety, vol. 204, 2020.

[25] H. Ziqing, and X. Liu, “Research on Software Reliability Growth Model Based on Gaussian New Distribution,” Procedia Computer Science, vol. 166 pp. 73-77, 2020.

[26] K. Sheoran, P. Tomar, and R. Mishra, “Software Quality Prediction Model with The Aid of Advanced Neural Network with HCS,” Procedia Computer Science, vol. 92, pp. 418-424, 2016.

[27] P. Sharma, “Software Quality Prediction using Hybrid Approach”, International Journal of Computer Applications, vol. 180, no. 4, 2017.

[28] I. Chawla, S. K. Singh, “An Automated Approach for Bug Categorization using Fuzzy Logic, India Software Engineering Conference, pp. 90-99, 2015.

[29] Deepika, O. P. Sangwan, “Neuro-Fuzzy Based Approach to Software Reusability Estimation,” vol. 9, pp. 151-159, 2016.

[30] R. Jose, “A Theoretical Framework for the Maintainability Model of Aspect Oriented Systems,” Procedia Computer Science, vol. 62 pp. 505-512, 2015.

[31] A. Okutan, O. T. Yıldız, “Software Defect Prediction using Bayesian Networks,” Empirical Software Engineering, vol. 19, no. 1 pp. 154-181, 2014.

[32] J. S. Challa, A. Paul, Y. Dada, V. Nerella, and P. R. Srivastava, “Integrated Software Quality Evaluation: A Fuzzy Multi-Criteria Approach”, Journal of Information Processing Systems, vol.7, no. 3, 2011.

[33] P. K. Singh, O. P. Sangwan, A. P. Singh, and A. Pratap, “A Quantitative Evaluation of Reusability for Aspect Oriented Software using Multi-criteria Decision Making Approach,” World Applied Sciences Journal, vol. 30, no. 12, pp. 1966-1976, 2014.

[34] N. Juneja, K. Upreti, “Software Quality Prediction in Aspect-Oriented Software by using Genetic Fuzzy System”, DIAS (Delhi Institute of Advance Studies) Technology, vol. 12, no. 2, 2016.

[35] P. Kumar, “Aspect-Oriented Software Quality Model the AOSQ Model”, Advanced Computing: An International Journal, vol.3, no.2, pp. 105-118, 2012.

[36] U. Chauhan, S. Sagar, “Analysis of Aspect Oriented Software Quality (AOSQ) Model,” International Journal of Advanced Research in Computer Science & Technology, vol. 3, no. 2, 2015.

[37] F. Amin, A. K. Mahmood, and A. Oxley, “Relative Importance of Factors Constituting Componen Reusability,” Academic Journals Inc. pp. 118–131, 2012.

[38] S. Goel, and A. Sharma, “Neuro Fuzzy based Approach to Predict Component’s Reusability,” International Journal of Computer Applications, vol. 5, pp. 33–38, 2014.

[39] P. K. Singh, O. P. Sangwan, and A. Srivastava, “An Essence of Software Maintenance Prediction using the Fuzzy Model for Aspect Oriented Software,” Asian Research Publishing Network, vol. 9, no. 9, 2014.

[40] M. Kassab, O. Ormandjieva, and C. Constantinides, “Providing Quality Measurement for Aspect-Oriented Software Development,” Asia-Pacific Software Engineering Conference, IEEE, 2005.