Journal of Advanced Computer Science & Technology, 4 (1) (2015) 60-67
www.sciencepubco.com/index.php/JACST
©Science Publishing Corporation
doi: 10.14419/jacst.v4i1.4211
Research Paper

Taxonomy of intelligence software reliability model

Saeed Ahmadluei

PhD. Student, Department of Computer Engineering, Islamic Azad University, Qazvin Branch, Qazvin, Iran
E-mail: saluei@qiau.ac.ir

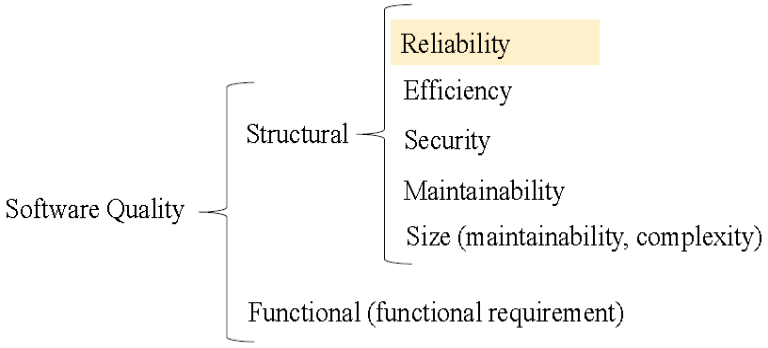
Abstract

The probability of failure free software operation for a specified period of time in a specified environment is called Reliability, it is one of the attributes of software quality and study about it come back to 1384. Exposition and spreading of new software systems and profound effect of it to human life emphasize the importance of software reliability analysis, until it poses formal definition at 1975. First race of reliability analysis methods that we called classic methods has stochastic process approach and in this way, attempt to predict the software behavior in future. Due to the ambiguity in fruitfulness of these solutions the challenge about reliability analysis continued till now. Great tendency in applying intelligence systems at variety of applications can be seen at 90 decade, and software reliability attracts some research direction to itself. Until now variety of methods in reliability analysis on the base of intelligence systems approach exhibited. In this survey the taxonomy of these methods represented with brief description of each one. Also comparison between these methods can be seen at the end of survey.

Keywords: *Genetic Algorithm, Intelligence System, Neural Network, Software Engineering, Software Reliability*.

1. Introduction

The increasing development of using software in sensitive and costly fields such as military systems' navigation, astronaut robots, medical subjects, many other various areas, and the growing complexities of productive applications clarify the necessity of presenting some approaches to evaluating the error-proof performance of applications along with the time and expenses spent in this area more than before. Reliability is the most important parameter of software quality in software engineering [7]. Its publicly accepted definition is as follows, "The probability of operating without failure during a specific period of time and in a specific environment."[5].

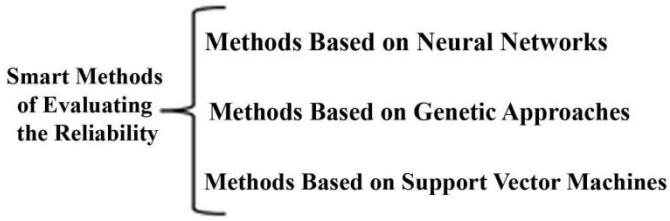


The history of evaluating the reliability of systems is traced back to 1384 [1]. Therefore, the subject was officially defined in software in 1975 [2]. This definition and the ones presented after it have not resulted in an accepted solution in this field so far [4]. The use of Intelligence methods has started since early 1990 in this field [3], and this new

approach peaked in the 90s, although some limited numbers of new papers are still presented in this field. Despite the fact that it appears unreasonable to evaluate software reliability without considering the hardware infrastructure, the hardware infrastructure is assumed to be flawless in the majority of models presented to evaluate software reliability. However, some models have been presented without this presumption to deal with the problem in combination [6].
A general classification of Intelligence methods for this field is presented in the second part. The third part deals with the methods pertaining to the neural networks used in this field. The fourth part investigates genetic algorithms, while the methods of support vector machines are studied in the fifth part. In the sixth part, some criteria are introduced to compare different methods, and the comparison of these methods and conclusion are presented in the seventh and eighth part, respectively.

2. Classification of Intelligence methods to evaluate the reliability

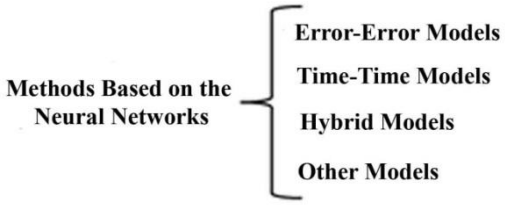
Generally, the Intelligence models of evaluating the software reliability are called nonparametric models. This is due to the performance of classic models as they are called parametric models [8]. Parametric (classic) approaches which have been presented to estimate the unknown parameters in the distribution function of the model led to the selection of the name parametric. In these models, functions named average functions or hazard functions which have one or more unknown parameters are presented. Using estimation methods, the values of these parameters are estimated, and then the value of reliability pertaining to future times is estimated by using the resulting function. The Intelligence models presented in this field can be classified into three general categories according to intelligent techniques and performance:



As mentioned earlier, this classification is done according to the approach which has been used; however, it is possible to do more detailed classifications in each group. This matter will be dealt with in the next parts.

2.1. Methods based on neural networks

It is obvious that many parameters such as the methodology used to develop the software applications, the software type, developing environment, software complexities, organization, the personnel producing the software application, and so forth influence the software reliability. Factors which are quite effective on the subject and the fact that these attributes focus on quality (so it is not possible to measure some of them precisely and numerically) have made the reliability face with a totally non-linear pattern. Dealing with such problems which are accompanied by many vague factors (in terms of quantity evaluation), the majority of experts would select neural networks as an appropriate approach because these networks are capable of estimating complicated functions quite well. Therefore, the neural networks have been paid more attention than two other techniques in this field, and many papers have been presented on this approach so far [9-15]. The neural networks were first presented in [14], [15] for this field.
Given the approach used in different papers which apply the neural networks, it is possible to classify these papers into four main groups as follows:



The first category includes error-error models. This appellation refers to the fact that we encounter models which gives the desired neural network the number of errors occurred during the previous tests and predicts the number of expected failures in the next interval. In other words, the inputs and outputs of the neural network is the number of errors [16], [17], [18]. Various papers reported different results as the type or structure of network changes and the number of neural network inputs varies. Using the multilayer feed forward network, recurrent network, and radial-based function

network, a comparison is made in [14] according to the square errors in future prediction. Table 1 indicates the results of this comparison.

Table 1: The Results of Comparing Three Neural Networks in Error-Error Group

	RMSE	
	Training Data	Test Data
MLP	0.6061	0.6677
RBFN	1.6465	0.1591
Elman	0.1625	0.1394

As it is observed, the performance of recurrent network has been reported to be better than that of the other ones. The higher capabilities of recurrent networks in predicting the parameters pertaining to the reliability have also been reported in comparison with other networks in other papers. In fact, this higher capability results from the innate ability of such networks in predicting real subjects [19], [20], and [21]. The poor performance of radial-based function neural networks is among the interesting reported results. Given the strength and flexibility of these networks in estimating the functions, this weak output can be caused by two factors: 1- The number of training data has not been sufficient, 2- The number of neurons has been small in the hidden layer of radial-based function network. Conducting other tests, if we, however, can prove that none of the above-mentioned reasons has not caused the poor performance of radial-based function neural networks, then the lack of input parameters (another one except for the number of failures) to estimate the output will be the only reason for this matter. This is a very important problem which has not been taken into account so far.

The second category includes time-time models. Like the first category, this one is named according to the type of input and output expected by the neural network. In this category, we encounter the models which give the times pertaining to the history of the application and those between failures to the neural network and predict the time to failure then. The papers proposed in this category are more than those of other categories. The reason can be sought in acquiring a rather satisfactory result in this category because no certain reason has been proposed so far [22-26]. A quite comprehensive investigation pertaining to the ability of multilayer feed forward network is done in [22] with the approach proposed in this group. However, no definite result has been presented for the structure of the optimized neural network in order to evaluate the reliability. In fact, the results of simulations conducted in [22] indicate that it is not possible to present such structure. In [22] and the other final results obtained by the authors of different papers in [22-26], the predictability mainly depends on the type of training data or the so-called input data. In other words and according to different papers [22-26], it is not possible to present a comprehensive neural network which has a constant ability while encountering different datasets.

It appears that the most interesting and useful approach to using the neural networks in order to predict the reliability is the hybrid models. However, these models do not have anything special by nature, and whatever they present is adapted from a viewpoint of classic models. In some cases in these models, the number of failures occurred during the previous intervals and sometimes the intervals pertaining to the previous periods are given to the neural network as the input. Likewise, the next interval or the number of the following failures is predicted. It is obvious that they are not any different from the other two categories; therefore, what makes these models different from the previous two categories is the combination of some models which evaluate the reliability of classic software in a neural network and present a hybrid output. Selecting the activity function intelligently for mid-layer neurons in these models, the performance of the neural network changes so that it appears to present a hybrid of classic models.

Like the classic models, the hybrid models attempt to find the unknown parameters of the assumed distribution function. Therefore, the difference is that the classic models attempt to find the unknown parameters by using the estimation methods (mostly maximum likelihood); however, the hybrid model of the neural network attempts to the appropriate values of weight coefficients which are the same as the unknown parameters of the distribution function by using the back propagation training technique or increasing the average error squares.

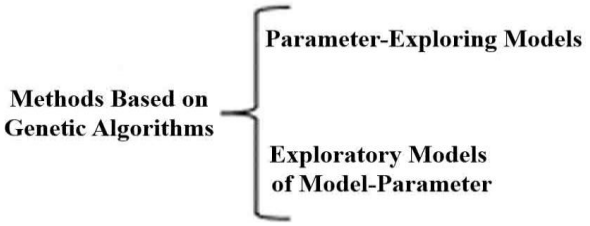
Therefore, the activity functions of different neurons can be selected according to the proposed distribution function in the classic models or by combining the output of these neurons in the next layer of the neural network so that a weighted output of majority is obtained from the classic models. An instance of these models is presented in [27] in which a comprehensive investigation and comparison of the performance of the hybrid model has been conducted on the usual neural network and the classic models, and a higher performance has been observed for the hybrid models.

Given the performance of the hybrid models of neural networks, we may be able to resolve the main problem of software reliability which is the selection of an appropriate model according to the environment and the target software application. In other words, selecting the activity function according to the distribution function proposed in the classic model and then the training network model and finally considering the weight coefficients calculated in the output layer, we attempt to select an equal classic model which has the maximum weight coefficient in the output layer.

There are a few models of the neural networks which do not fit into any groups according to the presented classification definitions, and they are not studied here due to the fact that they are limited and not very well-liked. The inputs and outputs of the neural networks are the intervals between failures and the number of failures predicted for the next interval or vice versa, respectively [18-29].

2.2. Methods based on genetic algorithms

The genetic algorithm is a method for searching in the problem space and finding the optimal value for the problem. Given this definition and the problem of estimating the unknown parameters of distribution functions in predicting software reliability in classic models, the way of using the genetic algorithms is clarified in this area. As it was stated earlier, the problem of estimating software reliability has turned into the problem of estimating the unknown parameters existing in the distribution functions; therefore, the problem can be turned into finding the optimal value for these parameters simply and by defining the parameters relating to a genetic algorithm. Although it is possible to present such a method, no actions have been taken so far, and no paper has discussed this matter. Perhaps the reason for is the lack of a valuable classic model or society which is publicly acclaimed (and therefore, it would be justifiable to spend time finding the optimal values of its parameters). Nevertheless, for the sake of classification integrity, the models of evaluating the software reliability which are based on genetic algorithms fall into two general categories:



The first group named parameter-exploring models refer to the models explained in specified classification. The second group named exploratory models of model-parameter includes the models which attempt to find the distribution function and the relevant parameters simultaneously. In other words, the approaches have been presented to search the space of functions by considering the sample data in order to select the best function in predicting software reliability. Given the performance of these models, they have been titled as exploratory models of model-parameter.

These models do not have any presumptions for distribution functions and attempt to find the function itself with the relevant parameters. According to the training data, we attempt to find the best distribution function (or the best function which can present an estimate for future, according to the training data). In other words, the problem of evaluating the software reliability has turned into an optimization problem to find the best function. Given the problem, it is obvious that the solution is genetic programing [30], a branch of genetic algorithms in which the individuals are the functions, and the operators which function on the individuals produce function, too. An instance of these models is presented in [31] so that the inefficiency of classic models is completely obvious in encountering some datasets according to the results. However, the Intelligence models indicate more flexibility, and they did not have disappointing results regarding any of datasets. Also, it has been clarified during the tests conducted in [31] that trigonometric and exponential functions are not efficient in this field. A very important result which has been referred to in [31] indicates the inefficiency of a certain model in encountering different datasets. This result is consistent with those of other models. In other words, the characteristics of input data have a great impact on the output of the proposed models, so it is not possible to select a special model as an efficient and comprehensive model to encounter every type of dataset. Statistically, the outputs of genetic algorithm are, however, better than those of other models. Therefore, the relative superiority of model-parameter exploratory genetic algorithm is clarified in comparison with the model of neural network and classic model. Perhaps the reason can be found in non-presumption approach of genetic algorithm models.

2.3. Methods based on support vector machine

The support vector machine [32] is used comprehensively to predict non-linear problems. The support vector machine has mainly presented for pattern recognition. However, its modified type named support vector regression (SVR) [33] has been presented. It is used to estimate the function or regression, in other words. The success of support vector machine in different fields has drawn experts' attention to software reliability. However, its usage has not been accepted for software reliability in comparison with other techniques. In fact, all the methods proposed in the field of software reliability have used the modified version of support vector machine named support vector regression. The main idea of this method is to attempt to find a function in order to estimate the number of failures or the interval between the next two failures. In other words, a classification can also be presented according to the type of input or output used like in neural network models. However, since the number of models presented with this approach is small, such classification has not been presented. According to the simulations conducted in the presented papers, it has been claimed that the results of predictions carried out by SVR are better than the results of genetic algorithms or those of neural networks [36], [34-39]. However, lack of reception of SVR in reliability in comparison with other methods takes an aura of mystery on this claim.

For instance, a model which uses data presented in [35] is proposed in [34]. It uses the cumulative time between two failures from the previous periods as the input, and the cumulative time between two failures pertaining to the next step

will be predicted. The interesting point in the results of [34] is the increased number of errors in the model based on support vector machine in comparison with neural network models as the number of previous input data increases. Given these results and also the rather satisfactory results of classic models based on Markov's model, it may be stated that predicting the next step regarding the cumulative time of failure does not depend on the all previous models. It is obvious that this matter is still a theory, and it has not been investigated or proven precisely so far.

3. Comparison of methods

The classic models state some presumptions on the environment and targeted application in the first place; therefore, they narrow down the application area of the model to simplify the presentation of the regulations over the model. In the first step in intelligent methods, we encounter the fact that these models have no presumption regarding the environment or the software application. This matter accounts for the main privileges of the intelligent models. Therefore, almost all the papers, in the majority of approaches, (except for the hybrid approaches of neural networks whose models are few) have reached the conclusion that the efficiency of the proposed model is highly sensitive to the input data. This issue refers to the inefficiency of Intelligence models while encountering all circumstances. In fact, it confirms the necessity of some presumptions which are stated in classic models. Therefore, two main flaws which all intelligent modes have are as follows:

1) Their disintegration or, in other words, their inability in presenting the satisfactory result in all environments and circumstances.
2) Lack of presenting the presumptions or the necessary circumstances for the satisfactory performance of the proposed model.

In Table 1, a general comparison of the explained methods is presented. As it was observed, three intellectualizing approaches which have been taken into account in software reliability field are neural networks, genetic algorithms, and support vector machines. These three approaches have been compared with each other in Table 2.

4. Conclusion

It is obvious that generating a flawless system is not possible; therefore, we cannot consider the objective of evaluating software reliability to be the production of flawless applications. Thus, the objective is to decrease the errors. So the answer to the question which asks, "What is the acceptable threshold of error in systems?" can resolve the challenge existing in the subject of evaluating the software reliability.

The classic models of evaluating the reliability mostly attempt to find a probability distribution function based on the subject so that they can predict the future according to that function. Therefore, finding this distribution function and predicting the future have turned out to be a challenge.

Intelligent approaches have not made special innovations in the main problem; however, they attempt to find the solution by accepting the problem the way it is (inputs, outputs, and assumptions). Also, the degree of this reception varies from a maximum value in the hybrid approaches of neural networks or parametric models of genetic algorithm to a minimum value in the exploratory genetic algorithms of the model. High dependence of all existing approaches (intelligent/classic) on the input data creates this theory that there are other efficient parameters which have not been taken into account for the definition of the main problem so far. For instance, the hypothesis which states no errors are added in the process of resolving the discovered error is totally different from the real world of software. This issue is simply overlooked in the majority of models.

Given the comprehensive researches which have been conducted in the evaluation of reliability classically and the still-remained challenge, it appears that accepting the existing problems (the way they are) and presenting intelligent approaches for them do not influence the problem solving so much. Moreover, the results of intelligent approaches proposed confirm this assertion. Considering the power of intelligent methods in presenting an approach for the problems in multi-dimensional spaces, it is expected that a step be taken in order to achieve this goal by redefining the problem of evaluating the reliability from a different perspective.

Finally, although accepting the input parameters of a problem which was introduced many years ago and using the modern techniques may sometimes be troubleshooting, it is not a new and terrific subject. The intelligent techniques are also considered to be troubleshooting to some extents in the field of software reliability. However, lack of precise and mathematical analysis of these techniques is a black point in this field in order to encourage the experts to use such techniques.

Table 1: The Presented Method Comparison

Method	Method Main Idea	Advantages	Disadvantages
Neural Network, Error-Error	According to the history record of the existing errors, it attempts to predict the errors in the future. The Errors are considered to be normal or cumulative.	• Simple Implementation • Different Datasets for training and testing the network	• Low predicting power which appears to be due to the independency of errors on each other in each period. • Low reception of this approach in comparison with two other neural networks
Neural Network, Time-Time	According the history record of times between the existing failures, it attempts to predict the time between failures in the future. The time may be cumulative or noncumulative. Also, it is possible the calendar time or the software runtime in the CPU may be considered.	• Simple Implementation • Availability of Datasets Required for Training and Testing the Network • Satisfactory results in various tests, although observing these results has not been proven while encountering other datasets with mathematic logic.	• Referring to the satisfactory results obtained from datasets under the test and not proving the desired efficiency of the proposed approach officially and systematically. • Make reliability depending on time while time refers to system execution time and test time.
Hybrid Neural Network	Presenting a special neural network, it proposes the hybrid behavior of classic models and attempts to estimate the output according to averaging or the majority.	• An appropriate mathematical base and lack of black box colliding with the neural network • Selecting an appropriate classic model in different environments, and therefore solving the problem of model selection in the appropriate environment • Benefiting from the majority in order to predict about future	• A few presented models make it hard to make remarks on them with this approach. • Whatever is presented is somehow adapted from other methods, and don't have anything but result combination or value making in their natures.
Genetic Algorithm, Exploratory-Parameter	Using the main ability of the genetic algorithm, it attempts to find the optimal value of unknown parameters in the functions with the functions proposed for classic models.	• Simple Implementation • The main characteristic of genetic algorithm which is the search for finding the optimal value is used.	• It cannot be considered as a single model. In fact, it is a method to find the unknown parameter of another model. Therefore, the method changes, and the value of unknown parameter and the output results will be different.
Genetic Algorithm, Exploratory of Model and Parameter	Using the genetic programing, it attempts to find the optimal function for evaluation and predicting the reliability.	• It solves the problem without any presumption. • Unlike other models, it assumes both the function and its parameters; therefore, it appears that it is more flexible while dealing with different datasets.	• The number of presented models is almost few; therefore, it is hard to make an absolute statement about this matter. • Achieving the desired result requires almost a lot of training data in comparison with other methods.
Support Vector Machine	Using SVR version of support vector machine, it seeks to estimate the evaluation function of reliability in the future.	• SVM is of the very efficient methods of estimation non-linear functions. • Compared with the models of neural networks, it has a high power of generalization; therefore, it indicates rather satisfactory results for different datasets.	• Compared with other methods, fewer models have been presented by this model, and this problem makes the efficiency evaluation difficult.

Table 2: Comparison of the Proposed Approaches

Approach	Main Idea	Advantages	Disadvantages
Neural Networks	Predicting the next value of time with the number of failures according to the previously available data.	• Simple Implementation • The efficiency of neural network has been proven as an estimation method of functions [40]. • It doesn't require a certain parameter adjustment. • It doesn't state a certain presumption to solve the problem. • Various available models according to this method simplify the evaluation and comparison. • An almost good ability in dealing with noise-making data.	• Dealing with the network in the form of black box, and the lack of precise and mathematic analysis regarding the efficiency of the proposed model. • They attempt to learn the model existing in training data, and they experience over fitting or lack of generalizability as usual.
Genetic Algorithm	Finding the Unknown Parameter or Functions to Predict the Future.	• Simple Implementation • Dealing with the problem without presumption	• It requires more training data in order to achieve the satisfactory results in comparison with other methods.
Support Vector Machine	Finding the Unknown Function in order to Predict the Future.	• High Generalizability • Low Error Tolerance While Dealing with Different Datasets	• Few available samples make the comparing evaluation difficult with this approach.

References

[1] Dionysius lardner "Babbage s' calculating engine" *computer society, 1834.*

[2] Johon D. Musa "A Theory of Software Reliability and its Application" *IEEE Trans. Softw. Eng., vol. SE-1, no.3, pp. 312–327, Sep. 1975* http://dx.doi.org/10.1109/TSE.1975.6312856.

[3] N.Karunanithi, D. Whitley "Prediction of Software Reliability Using Connectionist Models" *IEEE Trans. Softw. Eng., vol.18, no.7, pp. 563–574, Jul. 1992* http://dx.doi.org/10.1109/32.148475.

[4] J.P. Carnegie Mellon university http://www.ece.cmu.edu/~koopman/des_s99/sw_reliability/

[5] M.Xie,Y.S.Dai, K.L.Poh "Computing System Reliability Models and Analysis" *Kluwer AcademicPublishers – 2004*

[6] Lai, C.D., Xie, M., Poh, K.L., Dai, Y.S. and Yang, P. "A model for availability analysis of distributed software/hardware systems" *Information and Software Technology, 44 (6), 343-350. 2002* http://dx.doi.org/10.1016/S0950-5849(02)00007-1.

[7] R.Presman "Software Engineering A Practitioner`s Approach" *7edition, Mc Graw Hill,* 2010

[8] E.O.Costa,A.T.R.Pozo "A Genetic Programming Approach for Software Reliability Modeling" *IEEE Trans. Reliability, vol.59, no.1,pp. 222–230, Mar. 2010* http://dx.doi.org/10.1109/TR.2010.2040759.

[9] K.Y. Cai, L. Cai, W.D. Wang, Z.Y. Yu, D. Zhang, "On the neural network approach in software reliability modeling" *The Journal of Systems and Software vol.58, 2001, pp. 47–62* http://dx.doi.org/10.1016/S0164-1212(01)00027-9.

[10] T.Dohi, Y.Nishio, S.Osaki, "Optional software release scheduling based on artificial neural networks"*Annals of Software Engineering vol.8, 1999, pp.167–185* http://dx.doi.org/10.1023/A:1018962910992.

[11] N.Karunanithi, D.Whitley, Y.K.Malaiya, "Prediction of software reliability using neural networks" *International Symposium on Software Reliability, 1991, pp.124–130.*

[12] T.M.Khoshgoftaar, R.M.Szabo "Predicting software quality, during testing using neural network models:a comparative study" *International Journal of Reliability, Quality and Safety Engineering 1,1994, pp.303–319* http://dx.doi.org/10.1142/S0218539394000222.

[13] T.M.Khoshgoftaar, E.B.Allen, W.D.Jones, "Classification – tree models of software quality over multiple releases" *IEEE Transactions on Reliability vol.49, 2000, pp.4–11* http://dx.doi.org/10.1109/24.855532.

[14] N.Karunanithi, Y.K.Malaiya, "The scaling problem in neural networks for software reliability prediction" *Proceedings of the 3rd International IEEE Symposium of Software Reliability Engineering, Los Alamitos, CA, 1992, pp.76–82* http://dx.doi.org/10.1109/ISSRE.1992.285856.

[15] N.Karunanithi, D.Whitley "Prediction of software reliability using connectionist models" *IEEE Transactions on Software Engineering vol.18, 1992, pp.563–574* http://dx.doi.org/10.1109/32.148475.

[16] S.H.Aljahdali,K.A.Buragga "Employing four ANNs Paradigms for Software Reliability Prediction: an Analytical Study" *ICGST-AIML Journal, ISSN: 1687-4846, Vol.8, No.2, Sep 2008.*

[17] Aljahdali, S., Sheta, A., and Rine, D., "Predicting Accumulated Faults in Software Using Radial Basis Function Network" *Proceedings of the ISCA 17th International Conference on Computers and their Application, 4-6, April 2002, pp. 26-29.*

[18] Y.Wu, R.Yang "Study of Software Reliability Prediction Based on GR Neural Network" *School of Reliability and Systems Engineering Beihang University, DOI: 978-1-61284-666-8 IEEE 2010.*

[19] J. L. Elman "Finding structure in time" *Cognitive Science, pp. 179-211, 1990* http://dx.doi.org/10.1207/s15516709cog1402_1.

[20] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," *Proc. 8th Annual Conf. Cognitive Science, pp.531-546, 1986*

[21] R. I. Williams,D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation, vol. 1, pp. 270-280, 1989* http://dx.doi.org/10.1162/neco.1989.1.2.270.

[22] K.Y.Cai,L.CaiW.D.Wang "On the neural network approach in software reliability" *Journal of system and software(elseavier), vol.58, pp 47-62 ,2001* http://dx.doi.org/10.1016/S0164-1212(01)00027-9.

[23] N. Karunanithi, Y.K. Malaiya, D. Whitley "The scaling problem in neural networks for software reliability prediction" *Proceedings of the Third International IEEE Symposium of Software Reliability Engineering, Los Alamitos, CA, 1992, pp. 76–82* http://dx.doi.org/10.1109/ISSRE.1992.285856.

[24] N. Karunanithi, D. Whitley, Y.K. Malaiya, "Prediction of software reliability using connectionist models" *IEEE Tran. on Software Engineering vol.18, 1992, pp 563–574* http://dx.doi.org/10.1109/32.148475.

[25] R. Sitte "Comparison of software-reliability-growth predictions: neural networks vs parametric-recalibration*" IEEE Tran. on Reliability vol.48 NO.3, 1999, pp. 285–291* http://dx.doi.org/10.1109/24.799900.

[26] L. Tian, A. Noore, "On-line prediction of software reliability using an evolutionary connectionist model" *The Journal of Systems and Software, vol.77, 2005, pp.173–180* http://dx.doi.org/10.1016/j.jss.2004.08.023.

[27] Y.S.Su,C.Y.Huang "Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models" *Journal of Systems and Software(elseavier),vol.80, pp. 606–615, 2007* http://dx.doi.org/10.1016/j.jss.2006.06.017.

[28] T.M. Khoshgoftaar, R.M.Szabo **"**Using Neural Networks to Predict Software Faults During Testing" *IEEE Trans. Reliability., vol. 45, no.3,pp. 456–462, Sep. 1996* http://dx.doi.org/10.1109/24.537016.

[29] John R. Koza "Genetic Programming" *MIT Press 1998.*

[30] E.O.Costa, A.T.R.Pozo "A Genetic Programming Approach for Software Reliability Modeling" *IEEE Trans.Reliability., vol. 59, no.1,pp. 222–230, Mar. 2010 http://dx.doi.org/10.1109/TR.2010.2040759.*

[31] V.Vapnik "The Nature of Statistical Learning Theory" *Springer,New York, 1995* http://dx.doi.org/10.1007/978-1-4757-2440-0.

[32] A.J.Smola, B.S.Lkopf "A tutorial on support vector regression", *Statistics and Computing, vol.14,pp.199–222, Nov 2004* http://dx.doi.org/10.1023/B:STCO.0000035301.49549.88.

[33] B.Yang, X.Li "A Study on Software Reliability Prediction Based on Support Vector Machines" *University of Electronic Science and Technology of China, IEEE, doi : 1-4244-1529-2/07, 2007*

[34] A. Wood, "Predicting software reliability," *Computer, vol.29, no. 11, pp. 69-77, 1996* http://dx.doi.org/10.1109/2.544240.

[35] P.F.Pai, W.C.Hong "Software reliability forecasting by support vector machines with simulated annealing algorithms" *The Journal of Systems and Software vol.79, pp.747–755, 2006* http://dx.doi.org/10.1016/j.jss.2005.02.025.

[36] H.Can1, X.Jianchun1, Z.Ruide "A New Model for Software Defect Prediction Using Particle Swarm Optimization and Support Vector Machine" *University of Science and Technology, Nanjing, China, DOI : 978-1-4673-5534-6/13, IEEE,2013*

[37] Z.Qiuhong "Research of Software Failure Prediction Based on Support Vector Regression" *The 2nd International Conference on Computer Application and System Modeling, 2012*

[38] C.Jin "Software reliability prediction based on support vector regression using a hybrid genetic algorithm and simulated annealing algorithm" *IET Software, Vol. 5, Iss. 4, pp.398–405, 2011* http://dx.doi.org/10.1049/iet-sen.2010.0073.

[39] L.Tian, A.Noore "Evolutionary neural network modeling for software cumulative failure time prediction" *Reliability Engineering and System Safety(elsevier), vol.87, pp 45–51, 2005* http://dx.doi.org/10.1016/j.ress.2004.03.028.

[40] Osaki, Shunji "Software Reliability Models" *Book Section, Stochastic Models in Reliability and Maintenance,pp 53-280, Springer Berlin Heidelberg, P.B.2002*

[41] Jelinski, Z. and Moranda "Software reliability research*", In: Freiberger W. (ed), Statistical Computer Performance Evaluation, New York: Academic Press, pp.465-497. P.B. 1972*

[42] P. K. Kapur "Artificial Neural Networks Based SRGM" *Springer Series in Reliability Engineering"Software Reliability Assessment with OR Applications", DOI: 10.1007/978-0-85729-204-9_7, 2011* http://dx.doi.org/10.1007/978-0-85729-204-9_7.

[43] Iyer, R.K, Lee, "Measurement Based analysis of software reliability" *In: Handbook of software reliability engineering McGraw-Hill, New York, pp 303-358, 1996*

[44] Musa, J.D., Iannino, A., Okumoto, K. "Software Reliability,Measurement, Prediction and application" *McGraw-Hill, New York, 1987*

[45] G.Paris, D.Robilliard,C.Fonlupt "Applying Boosting Techniques to Genetic Programming" *Universit´e du Littoral-C´ote d'Opale,Springer-Verlag Berlin Heidelberg pp. 267–278, 2002*

[46] S.Yamada "Software Reliability Models" *Tottori University,Book chapter,Springer-Verlag Berlin Heidelberg, 2002*

[47] R.H.Reussner, H.W.Schmidt "Reliability prediction for component-based software architectures" *Journal of Systems and Software, vol.66, pp. 241–252, 2003* http://dx.doi.org/10.1016/S0164-1212(02)00080-8.

[48] R. Chillarege, "Orthogonal defect classification" *in Handbook of Software Reliability Engineering, M. R. Lyu, Ed. New York: McGraw-Hill,1996, pp. 359–400*

[49] S.Dick, C.L. Bethel, A.Kandel, "Software-Reliability Modeling: The Case for Deterministic Behavior" *IEEE Trans. On Systems man and cyber, vol. 37, no.1,pp. 106–119, Janu,2007*

[50] A.L.Goel, K.Okumoto "GoelOkumoto Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures" *IEEE Trans. Reliability., vol. 28, no.3,pp. 206–212, Aug. 1979* http://dx.doi.org/10.1109/TR.1979.5220566.

[51] Osaki, Shunji "Software Reliability Models" *Book Section of Stochastic Models in Reliability and Maintenance,pp 53-280, Springer Berlin Heidelberg, P.B.2002.*