Maximized Composite Functions Based Optimized Software Reliability Growth Function For Reliability Prediction

Y. Geetha Reddy , Dr. Y Prasanth

Abstract: As the size and complexity of the software systems are increasing day-by-day, the software reliability is an essential research concern for both software developers and clients. Software reliability make sure that the software products are failure free and reliable for software testing and product deploymentphases. Also, software reliable models are the key metrics to find the software quality and customer satisfaction. A large number of software reliability models have been proposed in the literature to improve the software product quality and failure free. However, no single model is suggested to optimize the software faults with high prediction rate and minimum error rate. Also, as the number of software faults are increasing in size then these models are restricted to limited parameter constraints for reliability prediction. Since, the reliability of the software products may increase or decrease depending on the selection of optimal parameter estimation and optimization functions. To overcome these problems, a novel software reliability prediction model with maximized composite functions is designed and implemented on various software failure datasets to estimate the software reliability with high accuracy. In the proposed model, a novel optimization parameter is used to improve the decision making on the software reliability using the optimized software reliability growth function. This proposed growth function is based on set of maximized composite functions for parameter estimation. Experimental results proved that the proposed model has high software reliability prediction rate with less error compared to the existing software reliability models.

1. INTRODUCTION

Software reliability is considered as the most vital factor during the complete process of software development. Most of the existing software reliability techniques consider the software faults are mutually interdependent to find the software quality assessment. But, these assumptions are not realistic in software applications due to correlated faults and limited test resources. Software development frameworks have been greatly optimized in different applications, such as grid system, national defence system, air control system and consumer applications. As the size of the software applications and framework are increasing exponentially, different approaches have been proposed in the literature to improve the software quality and reliability. Some of the common assumptions on the software reliability modelscan be summarized as:

1) Each software program contains only one type of software fault to normalize the detection level in the software product.
2) Software faults are uncorrelated and not depend on one another.
3) All the detected software faults during the testing phase should be removed from the software product.

- *Research Scholar1, Professor2*
- *Department of Computer science and Engineering*
- *KLEF, Guntur District, A.P., India.*

Failure: It is an event that happens when the deployed service deviates from the original services. Here, a service fails either due to insufficient software requirements specification or lack of proper software design. The deviation in the service is termed as an error and the cause of an error is termed as fault.

Fault Classification: All faults can be categorized based on multiple views (or) conditions. Different types of software faults and it's causes are summarized below:

1) Software And Hardware Faults: These types of faults are occurred in the software and hardware applications to affect the software and hardware data.
2) Malicious And Non-malicious Faults: These type of faults are occurred due to network attackers with the malicious and non-malicious objective of causing harm to the software applications or network systems.

Types of Software Reliability Analysis: There are two types of software reliability analysis are used based on internal structure and reliability measures.

1) Black Box Reliability Analysis (BBRA)
2) White Box Reliability Analysis (WBRA)

1) Black Box Reliability Analysis (BBRA): BBRA models are used to measure the reliability of the entire software application while ignoring it's internal structure. These models are used after the software development or after the deployment of the software applications. In the BBRA models software reliability growth measures are used for the prediction of failure data in the software applications. Black box models are not suitable to measure the failure in the

component based software applications. Therefore Black box models are less suitable than the White box models for software reliability analysis.

2) White Box Reliability Analysis (WBRA): WBRA models are used to measure the reliability of the entire software application by using the internal structure of the software. Initially, WBRA models are used earlier phases of software development especially, in the design and implementation phases. They are used to identify critical components that have the major impact on the software reliability assessment.

Technique	Parameters		Data sources	Process
	Level of details	Parameter type		
Metrics	Component, System	Probability of failure	Run-time artefacts, Design-time artefacts, Late stage artefacts, expert knowledge	Analytical, Metrics, Measurement
SRGMs	Component, System	Constant failure rate, Failure intensity	User information, Run-time artefacts, Late stage artefacts	Measurement, Metrics, Analytical
Test Code Coverage	Component, System	Constant failure rate, Failure intensity	Run-time artefacts, Late stage artefacts	Analytical, Measurement
Bug Report Analysis	Method, System, Component	Probability of failure, Failure intensity	Similar system, Expert knowledge, User information	Metrics, Analytical, Measurement
Input Domain Technique	Method, System, Component	Probability of failure	Export knowledge, Late stage artefacts, Run-time artefacts	Analytical, Measurement
Application Log Parsing	Method, System, Component	Probability of failure	Run-time artefacts	Analytical, Measurement
Fault Injection	Component, System	Probability of failure, Failure intensity, Constant failure rate	Predecessor system, Late stage artefacts	Analytical, Measurement

Software reliability can be defined as the probability of fault free operations within a restricted time period. During the software testing process, all the defects of software are identified and eliminated without hampering other functionalities. Hence, the reliability of software also increases. Thus, software reliability can be predicted through implementation of various software reliability growth models. These models usually used information from the software testing phase. After thorough verification, these SRGMs models are implemented in different large-scale or small scale projects. The above mentioned software reliability growth models can be decomposed into two sub-categories, those are:- Parametric models:-The parametric models usually give emphasis on the characteristics of software defects and failures. It considers several statistical characteristics just like probability distributions of time among software defects. According to the data representation, the above mentioned parametric model can also be subdivided into two subcategories, those are:-

a) Time dependent models:- these types of models consider times among consecutive software defects throughout the testing phase. It is capable of predicting certain probability measures of time to the next software defect. There are large numbers of research works in this field and according to the best model following assumption can be made:- the elapsed time among failures is managed through the exponential probability distribution. It generates

b) a parameter which is proportional to the rest numbers of defects.

c) Nonhomogeneous Poisson process based models:- according to this kind of models, software failures within a limited time period are considered. These models have an objective to predict the total numbers of software failures within a restricted time after successful completion of the testing and debugging phase. Among various kinds of models within this category, the best model has the following assumption:- the cumulative numbers of failures is dependent upon a Poisson process. Hence, the estimated numbers o failures within a particular time interval after time t is directly proportional to the estimated numbers of unidentified defects within time t.

Both of the above two categories of parametric models is dependent upon a priori assumptions related to the behavior of software defects and the stochastic characteristic of software defects. Apart from these, they also show various predictive performances in case of different projects.

1. Non-parametric models:- These types of models usually use various machine learning approaches such as artificial neural networks, genetic programming, support vector machines in order to predict the software reliability. Such models never require any kind of previous

assumptions. On the other hand, these models depends open fault history data. According to an advanced and new nonparametric approach, the concept of isolation of concern among the long-term trend in case of reliability growth and the local behavior is followed. Another nonparametric approach involves infinite testing effort function. Apart from the above, another efficient nonparametric technique is based upon the radial basis function neural network in order to predict software reliability accurately.

Software reliability is assumed as one of the vital software characteristic. Therefore, unreliable software may be expensive for the users. Again, it may hamper the reputation of developers. All the software reliability models include the time among failures. In other words, the data which describe the time series is considered. Presently, search based approaches are implemented in order to detect a pattern in the failure data. These techniques are capable to predict the reliability of the system more accurately. The most widely used approaches are dependent upon support vector machines, markov models, artificial neural networks, genetic programming, etc. By analyzing the results of the evaluation phase, we can mention here that, the above mentioned models outperform all conventional models in terms of performance. We can conclude here that, not a single model is perfect to be implemented in all kinds of scenarios. By through survey, we can mention here that, various functions which are required by genetic programming models still need a lot of enhancements in terms of performance. Through the process of model selection, the appropriate software reliability estimation can be ensured. The above mention task is not so easy and simple although for software experts. We can find thousands of software reliability models which have been proposed since decades. The main reason behind the difficulties is variability of reliability growth behavior. Certain models perform perfectly in most of the project scenarios, but those are not always the best for all kinds of software projects. The overall performance of the reliability model usually depends upon its data. A single reliability model is considered to be efficient, when its assumptions are satisfied. In order to choose the best model, it is required to analyze the failure data and sound knowledge about the previously existing software reliability models. Furthermore, there exists not a single standard which can be implemented along with high confidence level. Therefore, the selection process of an appropriate software reliability model is very complicated task. Practically, software developers usually implement various software reliability models with the help of several existing criteria. All the results are compared in order to choose the best model. The model implementation and analysis needs both effort and experience of software professionals. All previously developed techniques are limited to certain models. Apart from this, there exists another severe issue that is, not a single model provides automated support in case of selection. Hence, various data mining approaches can be implemented in order to automate the selection procedure of software reliability models. The learning process of the models emphasizes on the accumulation experience in case of a particular database. But, learning at the meta-level requires experience in the performance of multiple databases. In order to resolve all of the above mentioned issues, machine learning approaches are implemented in order to provide automated selection mechanism. Some researchers developed an advanced framework which is dependent upon Decision-Trees (DT). It dynamically chooses and integrates different numbers of software reliability models. Another issue is detected here that is, it never considers search-based approaches. Some of the selection procedures involve the implementation of artificial neural networks and certain meta-features. The artificial neural networks produce no symbolic classifiers which are complicated to interpret. Meta-learning approach is considered as the most convenient way to automatically select the appropriate software reliability model for a particular project. Meta-learning technique is generic in nature and it can be implemented in order to choose in between various types of reliability models. Machine learning approaches are needed in order to automatically gather knowledge for the process of model selection. It considers several characteristics in case of a singing reliability base. By considering the about knowledge, a meta- classifier is produced and this classifier can be implemented in order to select the most appropriate reliability model in case of new software projects. The application of meta-classifier never needs pre-evaluation of several models. Again, it does not need to gather any knowledge from the process of model selection. It may decrease requirement of the requirements of software professionals. Software reliability plays an important role in the software industry. It provides details about software failure to both the customers and the developers. Hence, the prediction of reliability is necessary in the present day scenario. The reliability growth model provides a perfect estimation of the total numbers of defects which may arise in the coming days after the successful product delivery. Therefore, the above mentioned models have the responsibility to estimate the time of a particular software release. Apart from these, the above models usually uses previous data collected from the software testing process. The said data is actually collection of total numbers of defects detected within a particular time frame at the time of software testing process. The mentioned data has the responsibility to be implemented in a reliability growth model in order to predict the reliability of the product. Most of the SRGMs contain a specific parameter which is dependent on the total numbers of faults identified within the set of code. Let us consider a scenario where, parameter and the present number of faults are known, in that case the rest faults can also be predicted appropriately. Knowledge about the numbers of residual faults always assists to determine is the code is ready for delivery or not. Again, it

also gives idea about how much testing is required before delivery. Furthermore, it provides an estimation of the total number of faults which the customers can detect at the time of their use.

All software reliability growth models need two kinds of data as input, those are mentioned below:-

1. Defect data
2. Time of the occurrence of the defects in case of defect data.

There exist two broad categories of software reliability growth models, those are:-

1. Parametric reliability growth models
2. Non-parametric reliability growth models

The parametric models are the traditional and oldest techniques those are used to detect the software reliability accurately. The basic concept that is used in the said models usually depends upon different statistical techniques such as regression. Below are some popular parametric models:-

In the year 1981, Bev Littlewood et al. tried to modify and extend the oldest and traditional SRGM. They adapted the concept of MLE in order to carry out the process of parameter estimation. The instability issue of the previous models is resolved in the said approach. Yamada et al. considered the error identification scheme and detected the curve of number of errors in case of analyzed data. The mentioned curve is like alphabet “S”. Therefore, this model is also known as S-shaped model. Yoshihiro Tohma et al. understood the importance of residual software defects. Hence, they developed a structural technique for identification of the above faults. There are lots of parametric and non-parametric models developed which are out of scope of this paper.

2.RELATED WORKS

M. Zhu and H. Pham proposed a new software reliability model [1]. This model uses time-dependent defects identification and defects elimination. There is a general consideration for all conventional SRGMs which is given below:- “Defect is completely independent and it can be eliminated easily without any hassle.” But, the above consideration is not satisfied in all of the cases because of different factors like software complexity, programmer proficiency, organization hierarchy, and so on. In this piece of research work, they introduced a new software reliability model. They proposed model is efficient for fault dependent identification, imperfect fault elimination and maximum number of fault identification. The genetic algorithm is implemented in order to predictive model parameters. The presented model is compared with other existing traditional approaches in terms of mean square error, predictive ratio risk, predictive power and Akaike information. Total three numbers of datasets are gathered for the demonstration of the above proposed model. Fault dependent identification and imperfect fault elimination are two important standards which are used to identify maximum numbers of faults. In future, further research works may be carried out to determine the appropriate time for software release. Again, the arrangement strategy of multi-release software product can be planned perfectly. J. Wang and C. Zhang developed an advanced software reliability prediction model [2]. They used the basic concepts of a deep learning model which depends upon the RNN encoder-decoder. Various kinds of software reliability models have been introduced since years in order to assess software reliability throughout the software testing phase. Of course the above mentioned models are efficient for assessment of software reliability, but none of them is able to predict the numbers of defects in each and every test scenario. With the advancement of technology, present day software contains numerous numbers of functions and hence, its size also increases. Therefore, it is quite hard and complicated to assess software reliability accurately. Significant performance can be achieved with the help of deep learning neural networks. The above model helps to deepen the layer levels and it gathers all the training characteristics. In this research paper, a deep learning model that depends upon RNN encoder-decoder is implemented in order to predict the total numbers of software defects. By analysing the results of the evaluation phase we can mention here that, this model is much better than that of other conventional approaches in terms of performance. Again, the prediction error is the lowest as compared to all other methods. Apart from this, it results better consistency, robustness, accuracy and stability. Restricted amount of fault data are used in this research work. By using more types of fault data, the overall performance can also be enhanced significantly. Hence, the future work may involve inclusion of more types of fault data in order to improve the overall performance. The delay among the fault identification and fault correction is considered as vital factor throughout the software testing process. Here, they used fault identification data like training data in case of deep neural networks. Further works are needed to integrate the fault identification data and fault correction data together. Additionally, larger class of artificial neural network may be considered in order to construct an efficient and effective software reliability model. S. Wang, Y. Wu, M. Lu and H. Li proposed a discrete non-homogeneous Poisson process software reliability growth model that depends upon test coverage [3]. In order to achieve test coverage, they presented two different discrete nonhomogeneous Poisson process software reliability growth models. It uses both failure data and test coverage with respect to the numbers of test cases. The test coverage function (TCF) is considered as the most vital factor. Initially, they introduced a specific TCF that depends

upon beta function. After that, they introduced two numbers of discrete mean value functions by merging two different concepts, those are:- test coverage and imperfect debugging. At last, the presented TCF and MVFs are calculated and validated in case of two actual software reliability datasets. Through analysing the outcomes of evaluation phase, TCF and MVFs result much improved estimation and fitting than that of other traditional techniques. Now-a-days, all the software reliability growth models include certain extra information just like test coverage. The defects presented in the covered constructs of the software can be revealed during the process of software testing. In this piece of research work, the relationship among reliability and test coverage is analysed. Initially, they explained a discrete TCF according to the numbers of test cases that depends upon the beta function. After that they presented the traditional and extended discrete MVFs by combining the test coverage along with the method of imperfect debugging. The above proposed technique is evaluated validated by considering two software reliability datasets. These datasets are gathered from original projects. By analysing the outcomes of the evaluation phase, we can state here that, it shows enhanced reliability fitting and estimation power as compared to other traditional approaches. In future, further research may be carried out in order to include more numbers of real failure datasets. Again, the mathematical expressions are required to be simplified in order to reduce the computational complexity. J. Iqbal studied different software reliability growth models in order to make a thorough comparison of linear and exponential fault content functions for study of imperfect debugging situations [4]. The process of software testing has a prime objective to gain confidence for implementation of the software in different real world applications. Reliability is always considered as vital factor during the whole process of software development. Through the implementation of error detection and correction mechanism, the reliability of software increases gradually. There are large numbers of software reliability growth models have been introduced since last four decades. Most of these models are dependent upon non homogeneous Poisson process framework. In this piece of research work, thorough comparison is made in between the performance of the above presented technique along with other traditional techniques. In this comparison linear and exponential fault content functions are included in case of imperfect debugging. The performance of this technique is compared with certain previously developed approaches with inclusion of real world data sets. Three numbers of important metrics are used in order to evaluate the performance, those metrics are:- mean square error, predictive ratio risk and predictive power. In this research paper, they have proposed to advanced software reliability growth models with the help of two previously existing imperfect debugging techniques. The mean value functions of all of these methods are included in order to find estimated parameters. We can state here that, the presented model 2 always shows better performance as compared to model 1. Again, model 2 results enhanced predictive power as compared to Model 1 in case of four datasets. Further research works may be performed in order to extend the traditional forms of imperfect debugging. K. Rekab, H. Thompson and W. Wu introduced an effective test allocation approach for appropriate software reliability estimation [5]. They have introduced an advanced sampling technique in order to implement during the software reliability estimation process. On the other hand, the constant sampling techniques in which the proportions of the test cases are gathered from every individual partition and these proportions are calculated prior to the execution of reliability testing. The allocation decisions are performed dynamically during the testing phase. After that these estimates are refined iteratively. The outcomes of the above implemented sampling technique are compared with the best constant sampling technique. Here we can state that, the proposed technique shows better performance than that of best constant sampling technique in terms of estimated loss. In this piece of research work, they have implemented a Bayesian technique in order to perform the process of test case allocation. The above mentioned Bayesian allocation helps us to update the traditional considerations related to the reliability of every individual partition. Hence, dynamic refinements of test cases are performed throughout the reliability testing process. In this research paper, an accelerated sequential sampling technique is implemented in order to estimate the reliability of a software system with the help of partition testing process. In case of previously defined test case allocation, the above presented technique is far better as compared to the traditional constant sampling mechanisms. Here, they have considered that, for every individual sub-domain, the conditional reliability needs a beta distribution. R. Garg, K. Sharma, R. Kumar and R. K. Garg performed a detail performance analysis of software reliability growth models with the help of matrix approach [6]. In this research paper, they have proposed a computational methodology that depends upon matrix operation in order to create a computer based solution for performance analysis issues. There are a group of 7 comparison strategies developed in order to rank different nonhomogeneous Poisson process software reliability models. These models have been developed since last four decades in order to estimate various software reliability measures just like total number of defects, failure rate and reliability. It is necessary to choose the appropriate software reliability model and hence, it has become the most challenging research domain. All the traditional selection techniques to choose appropriate software reliability models are not efficient enough for high level of confidence. In the above scenario, restricted numbers of model selection characteristics are used. The original data set with medium size is considered for this research. The outcome of this work is a ranking strategy that

depends upon the Permanent value of the criteria matrix. In this piece of research work, the problems of performance analysis are considered seriously. The above proposed model permits the user to carry out a basic analysis according to his personal preferences. This technique is the best choice for ranking of different software reliability models that depend upon a numbers of confusing characteristics along with equal or unequal weights. A basic mathematical formulation is implemented in the above proposed technique. Matrix operation is needed to be executed through the computing devices efficiently. Most of the complex multi-attributes decision issues are resolved through the implementation of the above proposed technique. J. Pati and K. K. Shukla developed a hybrid approach in order to carry out the process of software reliability prediction [7]. Software reliability is considered as a vital factor during the whole process of software development. The exact estimation of software reliability is completely hard and complicated process. There are lots of software reliability models which are implemented during the prediction process. But, not all models are efficient and effective. Most of these models include numbers of unrealistic assumptions and they depend upon their surrounding environment. Apart from these, these models never achieve satisfactory accuracy. In this research work, they have implemented a time series technique in order to carry out the software reliability prediction process efficiently. Here, they have implemented an ensemble based approach known as hybrid ARIMA in order to predict software reliability depending upon real world failure data sets. A comparative analysis is performed in order to estimate the performance of hybrid ARIMA and other ARIMA models. By analysing the outcomes, we can state here that, hybrid ARIMA can enhance the prediction accuracy more efficiently[9][10]. There are large numbers of software reliability growth models. Not all of them are efficient because of unrealistic assumptions and poor predictability. The traditional ARIMA model is not also much efficient enough because it results very poor performance during the fitting of data. Therefore, the traditional ARIMA model needs further modifications and enhancements in order to improve the accuracy[11]. The above proposed hybrid ARIMA is actually the integration of the traditional ARIMA model with another Neural Network approach[12] in order to model the software failure interval series. It is basically data-oriented technique and considered as a complete solution for all issues. The predictive accuracy of the traditional ARIMA is compared with the predictive accuracy of hybrid ARIMA in order to detect the best model for software prediction process.

PROPOSED MODEL

In this section, a novel approach to predict the software reliability and quality through mean value function of the multi-criteria based model selection and optimization. The main idea of this approach is to estimate the software reliability with high accuracy using the learning algorithms. Fig 1, represents the overall approach. This approach first computes the software reliability factor using the proposed maximized optimization function for reliability efficiency.

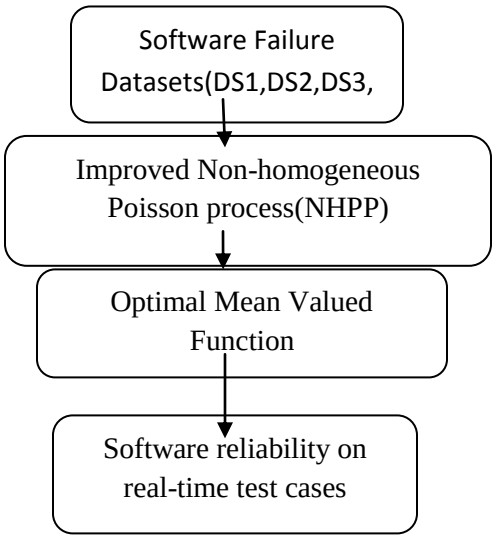

Figure 1: *Proposed Approach Overview*

Initially, different software reliability failure datasets are taken as input to the proposed framework. For each dataset, reliability function is applied to find the mean failure rate on the given input features and parameters.

Optimal Software Reliability Growth Function

In the proposed model, a novel reliability prediction measure is proposed to find the mean time to failure on the software failure data. The maximization function used to optimal mean time failure computation is given as

$$\phi = \mathrm{Max}\{\frac{1}{2} - \frac{1}{2}|\frac{\log(x) - \mu_{DS[i]}}{\sqrt{2}.\sigma_{DS[i]}}|, e^{-(\alpha * x)^{\beta}}, 1 - e^{-(\alpha * x)}\}$$

Here the maximization of the three reliability functions is used for fault estimation process. The mathematical function used to find the mean time fault detection process is given as:

$$F(t) = \log|\frac{\alpha}{1+\beta.e^{-\phi.t}}| ------(1)$$

where α,β,ϕ>**0**

$$f_1(x) = e^{-\phi.x}$$

$$f_2(x) = \frac{1}{1+\beta.x}$$

$$f_3(x) == \alpha.x$$

$$f_4(x) = \log|x|$$

Now, these functions derive mean value function as shown below:

$$f_4(f_3(f_2(f_1(x)))) = f_4(f_3(f_2(e^{-\phi.x})))$$

$$= f_4(f_3(\frac{1}{1+\beta.e^{-\phi.x}}))$$

$$= f_4(\alpha.\frac{1}{1+\beta.e^{-\phi.x}})$$

$$= \mathbf{log}|\alpha.\frac{1}{1+\beta.e^{-\phi.x}}| = F(x) ---(2)$$

Here, proposed maximization function in eq(2) satisfies the composite function of four functions.

Optimized Software Reliability Growth Measure:

Predicted number of failures for software growth reliability growth is given by

$$R_n = SS * F_d - N_r = p.qt^{q-1}$$

where t=time variant

SS = Software size

F_d = Fault density

N(t)=number of failures at time t

t_0 = first fault

$$k = \phi.\frac{(SS * F_d - N_r)}{SS * F_d}$$

$$q = \frac{-\log(1-k).SS * F_d}{\log(t_0) - \log(T)}$$

$$p = \frac{(1-k).SS * F_d}{T^q}$$

Experimental Results

Experimental results are carried out on the software failure datasets taken form the DS1 reported by Dr. K.Okumoto. During 56 weeks of testing, a total of 124 faults are identified to test the stability. The second , third and fourth datasets DS2,DS3,DS4 are taken from Rome air development center(RADC) projects.

DS1 Software Reliability Patterns:

[W=20]: 1 ==> [CF=100, Label=H]: 1
[W=20, CF=100]: 1 ==> [Label=H]: 1
[W=20, Label=H]: 1 ==> [CF=100]: 1
[W=19]: 1 ==> [CF=100, Label=H]: 1
[W=19, CF=100]: 1 ==> [Label=H]: 1
[W=19, Label=H]: 1 ==> [CF=100]: 1
[W=18]: 1 ==> [CF=100, Label=H]: 1
[W=18, CF=100]: 1 ==> [Label=H]: 1
[W=18, Label=H]: 1 ==> [CF=100]: 1
[W=17]: 1 ==> [CF=99, Label=H]: 1
[CF=99]: 1 ==> [W=17, Label=H]: 1
[W=17, CF=99]: 1 ==> [Label=H]: 1
[W=17, Label=H]: 1 ==> [CF=99]: 1
[CF=99, Label=H]: 1 ==> [W=17]: 1
[W=16]: 1 ==> [CF=98, Label=H]: 1
[CF=98]: 1 ==> [W=16, Label=H]: 1
[W=16, CF=98]: 1 ==> [Label=H]: 1
[W=16, Label=H]: 1 ==> [CF=98]: 1
[CF=98, Label=H]: 1 ==> [W=16]: 1
[W=15]: 1 ==> [CF=96, Label=H]: 1

DS2 Patterns:

[CF=164]: 2 ==> [W=14, Label=H]: 1
[CF=164, Label=H]: 2 ==> [W=14]: 1
[CF=164]: 2 ==> [W=13, Label=H]: 1
[CF=164, Label=H]: 2 ==> [W=13]: 1
[Label=M]: 2 ==> [W=7, CF=63]: 1
[Label=M]: 2 ==> [W=6, CF=37]: 1

[Label=M]: 2 ==> [CF=63]: 1
[Label=M]: 2 ==> [CF=37]: 1
[CF=164]: 2 ==> [W=14]: 1
[CF=164]: 2 ==> [W=13]: 1
[Label=M]: 2 ==> [W=7]: 1
[Label=M]: 2 ==> [W=6]: 1
[Label=L]: 5 ==> [CF=29]: 4
[W=18]: 1 ==> [CF=176, Label=H]: 1
[CF=176]: 1 ==> [W=18, Label=H]: 1
[W=18, CF=176]: 1 ==> [Label=H]: 1
[W=18, Label=H]: 1 ==> [CF=176]: 1
[CF=176, Label=H]: 1 ==> [W=18]: 1
[W=17]: 1 ==> [CF=170, Label=H]: 1
[CF=170]: 1 ==> [W=17, Label=H]: 1
[W=17, CF=170]: 1 ==> [Label=H]: 1
[W=17, Label=H]: 1 ==> [CF=170]: 1
[CF=170, Label=H]: 1 ==> [W=17]: 1
[W=16]: 1 ==> [CF=168, Label=H]: 1
[CF=168]: 1 ==> [W=16, Label=H]: 1
[W=16, CF=168]: 1 ==> [Label=H]: 1
[W=16, Label=H]: 1 ==> [CF=168]: 1
[CF=168, Label=H]: 1 ==> [W=16]: 1
[W=15]: 1 ==> [CF=165, Label=H]: 1
[CF=165]: 1 ==> [W=15, Label=H]: 1
[W=15, CF=165]: 1 ==> [Label=H]: 1
[W=15, Label=H]: 1 ==> [CF=165]: 1
[CF=165, Label=H]: 1 ==> [W=15]: 1
[W=14]: 1 ==> [CF=164, Label=H]: 1
[W=14, CF=164]: 1 ==> [Label=H]: 1
[W=14, Label=H]: 1 ==> [CF=164]: 1
[W=13]: 1 ==> [CF=164, Label=H]: 1
[W=13, CF=164]: 1 ==> [Label=H]: 1
[W=13, Label=H]: 1 ==> [CF=164]: 1

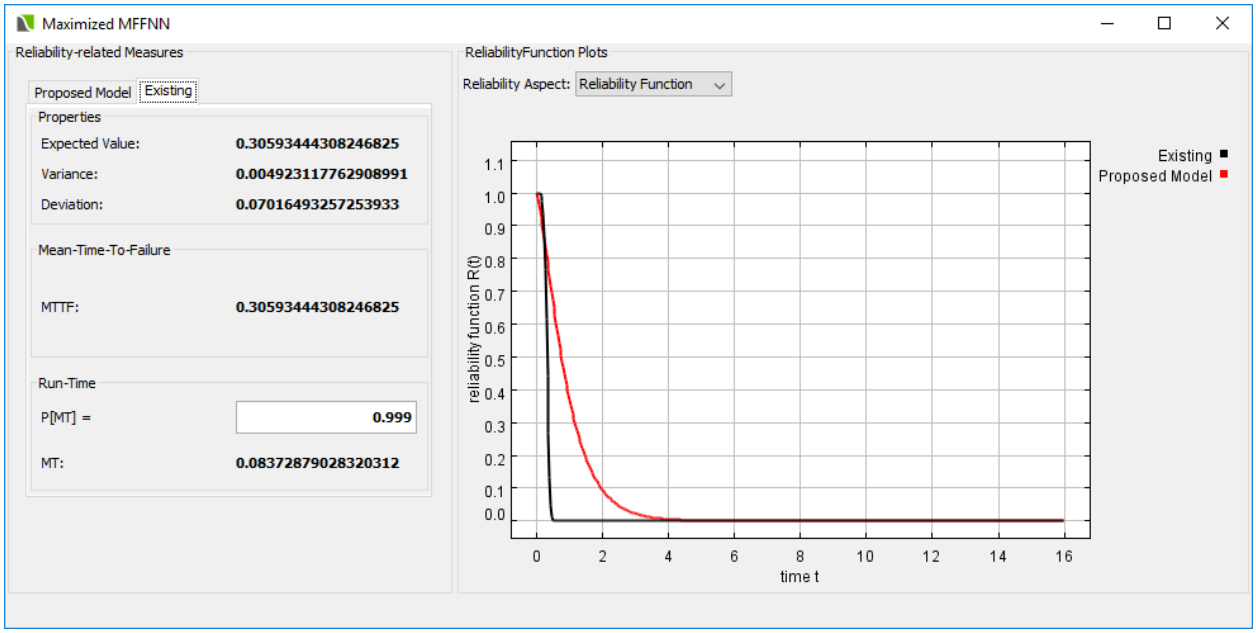

Figure 2: *Existing Approach*

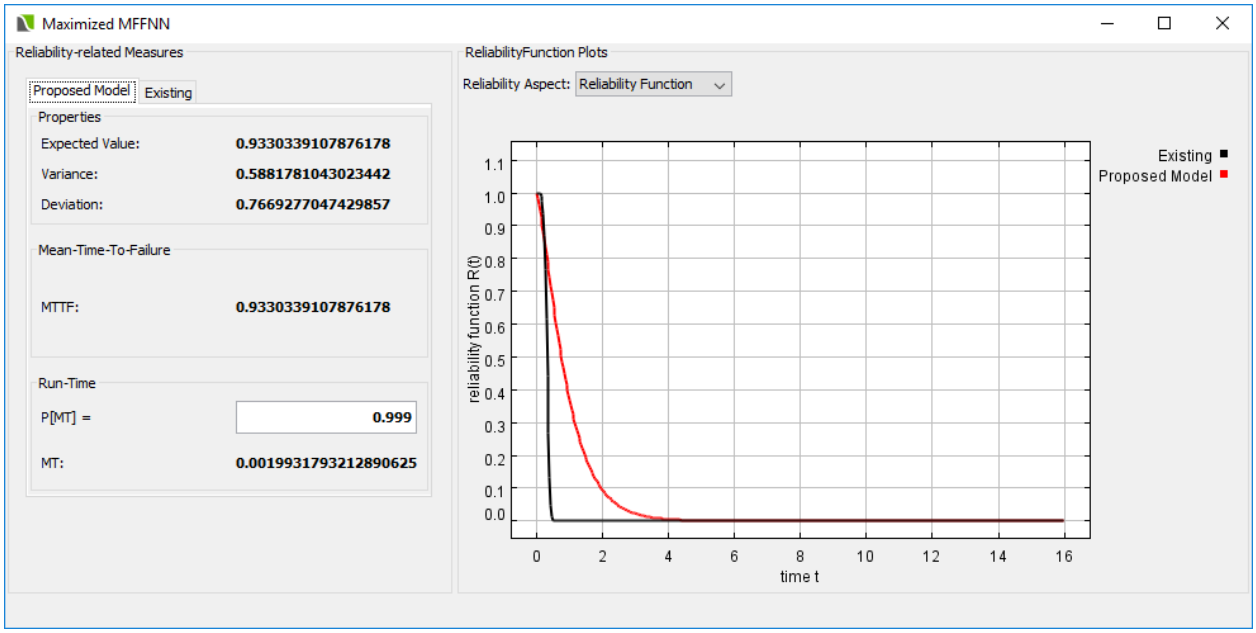

Figure 3: *Proposed approach*

Datasets	CASRE	RGA	Proposed
DS1	207.43	139.54	127.46
DS2	15.36	11.64	10.54
DS3	28.46	17.35	11.54
DS4	32.745	28.44	25.75

CONCLUSION

Software reliability is considered as the most important factor in the process of software development. We can mention here that, with increase in the software reliability, the quality of software also increases. At the time of testing and debugging, all defects are detected and separated carefully. Hence we can assume that, the reliability of software improves gradually with testing and debugging process. Software reliability can be predicted by various kinds of software reliability growth models. These models gather information from the software testing and debugging process. The above mentioned software reliability growth models can be decomposed into two sub-categories, those are:- 1) Parametric models and 2) Non-parametric models. Not all prediction models are efficient and effective for all kinds of projects. Therefore, there is necessity of some selection strategy in order to choose a perfect prediction approach for a particular case. To overcome these problems, a novel software reliability prediction model with maximized composite functions is designed and implemented on various software failure datasets to estimate the software reliability with high accuracy. In the proposed model, a novel optimization parameter is used to improve the decision making on the software reliability using the optimized software reliability growth function. This proposed growth function is based on set of maximized composite functions for parameter estimation. Experimental results proved that the proposed model has high software reliability prediction rate with less error compared to the existing software reliability models.

REFERENCES

[1] M. Zhu and H. Pham, “A software reliability model with time-dependent fault detection and fault removal”, “Vietnam J Comput Sci (2016) 3:71–79”.

[2] J. Wang and C. Zhang, “Software Reliability Prediction Using a Deep Learning Model based on the RNN Encoder–Decoder “.

[3] S. Wang, Y. Wu, M. Lu and H. Li, “Discrete Nonhomogeneous Poisson Process Software Reliability Growth Models Based on Test Coverage”, .Qual. Reliab. Engng. Int. 2012.

[4] J. Iqbal, “Software reliability growth models: A comparison of linear and exponential fault content functions for study of imperfect debugging situations”,”Cogent Engineering (2017), 4: 1286739.

[5] K. Rekab, H. Thompson and W. Wu, “An efficient test allocation for software reliability estimation”, Applied Mathematics and Computation 220 (2013) 94–103 .

[6] R. Garg, K. Sharma, R. Kumar and R. K. Garg, "Performance Analysis of Software Reliability Models using Matrix Method",World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering Vol:4, No:11, 2010

[7] J. Pati and K. K. Shukla, "A Hybrid Technique for Software Reliability Prediction", "ISEC '15, February 18 - 20, 2015, Bangalore, India" pp. 139-146.

[8] Yuyu Yin, Lu Chen, Yueshen Xu, Jian Wan:Location-Aware Service Recommendation With Enhanced Probabilistic Matrix Factorization. IEEE Access 6: 62815-62825 (2018)

[9] Yu, Jun; Tao, Dapeng; Li, Jonathan; Cheng, Jun.: Semantic preserving distance metric learning and applications. INFORMATION SCIENCES, 2014, 281: 674-686.

[10] Shen, Jing; Wan, Jian; Lim, Se-Jung; Yu, Lifeng.: Random-forest-based failure prediction for hard disk drives. INTERNATIONAL JOURNAL OF DISTRIBUTED SENSOR NETWORKS, 2018, 14(11).

[11] Yuyu Yin, Song Aihua, Gao Min, Yueshen Xu, Wang Shuoping:QoS Prediction for Web Service Recommendation with Network Location-Aware Neighbor Selection. International Journal of Software Engineering and Knowledge Engineering 26(4): 611-632 (2016)

[12] He Huang, Shufang Zeng, Arun Kumar Sangaiah, Jin Wang, Beamforming Aided SSK Modulation for MIMO System with Energy Harvesting, Journal of Intelligent & Fuzzy Systems, vol.36,no.5, pp.4017-4023,2019.