International Journal of Engineering & Technology, 7 (1.1) (2018) 397-400

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Bug reduction with reliability factors using hybrid reliable model

T. Vijaya Saradhi [1], M. Manisha [1], L.Vijaya Lakshmi [1], K.Gowtham [1] *

[1] *Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation*
**Corresponding author E-mail: gowtamkanneganti9@gmail.com*

Abstract

Software reliability is the probability that software will not cause the failure of a product for a specified time under specified conditions. Software Reliability is also an important factor affecting system reliability. Software reliability is different compared to hardware reliability. Hardware reliability is the one which deals with manufacturing something which is related to external features of the system. Reliability is one of the major problem faced by the systems. Hence, there are some factors which make the system much reliable. As there are different types of software reliable models we analyses each of its performance and also the factors which are affecting them. In this paper we propose a hybrid model which is used for the reduction in the count of bugs and also we implement the reliability factors for that model. Therefore, this model reduces number of bugs. As we know reliability is inversely proportional to the number of bugs and therefore by using this model we are increasing reliability.

Keywords: *Failure; Hardwarereliability; Reliability; Reliabilityfactors; Reliablemodels.*

1. Introduction

Software reliability is one of the important factor which effects the system and it can be defined as a condition where the system do not encounter any problems in a certain limit of time. Garima Chawlaet [1] states that programming unwavering quality is going to characterize the dependability or the life of programming framework with various properties as, we know there are different types of reliable models which can describe reliability. We may think how reliability can be one of the main cause for the systems which we are using in present situations. If the system is unreliable it even causes the data stored in the system to be lost. We can assume the loss in different ways and it is very difficult to make modifications for the system which is unreliable to make it reliable. Felipe Febrero,Coral [2] referenced that now a day's failure rate of the systems is being high. That not only includes in the field of software but also in other fields, this may cause heavy loss
Sometimes not only in the terms of amount but that may also include loss of lives. Hence that is the reason why reliability plays a major role. There are different factors which effect the system reliability. Zhu, M., Zhang, X., Pham, H [3] made a comparison using this factors. By combining both of them we get the best model as an output. In this paper, as we describe different models and study about each of them in a detailed manner. So, we can predict the best model and if we implement reliability factors then that becomes an efficient model and can be used in further processes. Therefore, this help us to give the clear idea about different types of models and their working using their functionalities. So, this helps the future models to be more reliable than the present one. For more reliability we can even modify different parameters such that it may give us a new model.

2. What are reliability factors?

As we know software usage is being increased day by day. If the system causes any failure that the correction becomes more complex and also it requires a lot of cost. Nisha Sharma[4] states that programming dependability is the pivotal factor to gauge the product quality and additionally to assess the product cost So, to reduce the cost and failures we have to know about the system and also the reasons why the system is being failed. Therefore, there are some factors which help the system without being failed and they are known as reliability factors. The factors are:

2.1. Program complexity

This is actually measured in terms of kilo line of code and if this factor is greater than 50 that is said to be high complex program else it is said to be low. This should be low because a complex program or system may cause different errors and that may lead to a complete failure of the model. As the number of errors increase the system or the model becomes more complex. Number of errors are always proportional to the program complexity.

2.2. Cost

This is a factor which plays a major role in any of the project because this optimization will lead the success of the project and makes it more reliable. There should be the correct estimation of cost and this also includes a correct planning. If a certain limit is attained to the cost the project should not exceed it. In-case if it is exceeded it should be informed. Therefore this is one of the most important factor.

2.3. Testing methodologies

As we know in any of the model testing plays a very important role and the methods which we choose for testing should be able to identify minimal errors also, because large errors can be solved but minor errors are very difficult to identify and may cause the system to fail. Most of the people assume that testing should be done only after construction phase but it is always advisable to perform testing from requirements phase itself. Hence different methods based on the phase should be choosen correctly and then applied which makes the system more reliable.

2.4. Less number of failures

Failure is defined as the condition where system will not work as it is required. So, if these are less the customer will be satisfied. The customer satisfaction is the key role of success in any project that may belong to any field. As we know errors are always inversely proportional to reliability. Errors are the main Cause for the failures in the system. So, to maintain a reliable system there should be less number of failures.

2.5. Documentation

This is a factor of reliability because any of the project can be understood by the correct documentation only. If there is an ambiguous document then that even leads to the confusion for the members in the team. Therefore to maintain a reliable system documentation is needed and that also plays a key role in development of further projects or software's.
Hence, there are some more factors but these are some of the important one's which make the system to be more reliable and comfortable.

3. Different types of reliability models

According to Amanpreet Kaur, Kailash Bahl [5] states that models are categorized into 4 types and they are:

1) Time between failures model
2) Failure count model
3) Fault seeding model
4) Input domain based model

3.1. Time between failure models

Razeef Mohd., Mohsin Nazir [6] states that this model estimates the time between failures. There are some assumptions for this model which include no faults should be introduced during correction, there should be independent time between failures, there should be equal probability of each failure. The models which come under this category are:

1) Jelinski- Moranda's model.
2) Shick and wolverton's model.
3) Goel-Okumoto imperfect debugging model.
4) D Littlewood-Verrall s Bayesian model.

3.2. Failure count model

This includes number of failures occurred in specific interval of time. Razeef Mohd., Mohsin Nazir[6] referred that failure counts are assumed to follow a known statistical random variables which are time dependent may be discrete or continuous failure rate There are some models which are included in this they are:

1) Shooman s exponential model.
2) Goel's generalized nonhomogeneous Poisson.
3) Musa s execution time model.
4) Musa-Okumoto's exponential model.

3.3. Fault seeding model

In this model, we assume that there are expected number of failures in the model. Amanpreet Kaur, Kailash Bahl[5] states that this is an essential approach in this class of models is to "seed" a known number of issues in a program which is expected to have an obscure number of indigenous faults. So, we plot number of failures in the model and observe whether the model is able to recognize the errors or not. This model includes:

1) Mill's error seeding model

3.4. Input domain based model

This model is estimated in a way that there are some test cases which are selected randomly from well-known inputs in the program. The model included in this is:
Nelson's model

3.4.1. Jelinski- moranda's model

This model was the earliest model in the reliability. There are some assumptions for this model to be considered

1) All faults in the program have the equal proportion to cause the error
2) The hazard rate is equal to the number of remaining faults in the program
3) No new errors or debugging should be introduced during testing in between the program
4) In the beginning it was assumed to remove only one failure for one time, but later it was modified that more than one faults can be removed in one time.
5) Errors may occur accidently

The rate of failure is proportional to remaining number of defects

$$Z(T) = Ø\,[N-n] \text{ where } n=0, 1, 2\ldots. \quad (1)$$

Ø = constant failure intensity produced by each failure
N= faults before testing done
This belongs to the class of exponential models. As this is one of the early models many other researches were based on this model taking this as a reference. This model is applicable in the cases of the above assumptions only. This model explains about the dependence of time taken place during testing phase. Due to more disadvantages and unrealistic assumptions this model is not in use for the present models.
Disadvantages:

1) This is a time consuming model.
2) The assumptions in this model are not realistic.
3) Many test cases cannot be solved using this model.
4) It is not suitable for large models.

As more faults cannot be removed it becomes a problem for every time testing

3.4.2. Musa-okumoto's exponential model

This model was proposed by Musa and Okumoto and this was also one of the earliest model. This model is based on the data which is failed and measured in execution time. It belongs to the family of geometric models and its type is poison. This model assumes infinite failures in the system.
Assumptions:

1) The software should be performed in such a way that its satisfies the reliability predictions which are made.
2) Every fault has an equal chance of occurring
3) When faults are detected failures are said to be independent.
4) When expected number of failures are experienced failure intensity will be decreased. Poisson process will be followed by the cumulative number of failures

$$\Delta(\mu) = (1/\Theta) \ln(\lambda x/\lambda y) \quad (2)$$

This model describes that repairing of first failure has more impact on program compared to other failures and also it decreases exponentially. This model do not have a direct effect on the reliability. In this model the failure rate will be decreased rapidly in the case of early execution. Based on this model we can get a clear idea that reliability depends on both execution time and also the failure intensity of the system. This models produces a good value for the reliability by reducing the number of failures.

3.4.3. Mills error seeding method

Amanpreet Kaur, Kailash Bahl [5] stated that seed is the fault in the program.In this model we seed the errors and check whether our model is performing in a right way or not. The main objective of this model is to find residual errors and how the software should be released after finding the results.
This includes two phases:

1) Detection of the error
2) Seeding of the error

In the first phase, we try to detect the errors which were present in the model while constructing and as there were many number of errors it is difficult to categorize them because it based on the Program. The results of the first phase will be successful in most of the cases. Based on the first phase the second one will be done.
Second phase includes seeding of the errors. Based on first phase results we seed errors in a normal distribution form.
Residual errors can be defined as the one which are to calculated and that were left over in the model. There are two types of errors which are mainly defined as inherent errors and induced errors. Inherent errors are the something which are not possible to avoid their occurrence, where user should take care of the errors and should modify them in each and every aspect. Induced errors are the one which are made intentionally. Razeef Mohd., Mohsin Nazir [6] stated that this model is used to assess software reliability by taking certain measures.

$$N= (n(r-k)/k)-1 \quad (3)$$

n= initial errors
r=number of errors removed during debugging
k=errors which are unavoidable
Disadvantages:

1) This is a very costly method to perform reliability
2) Cannot be applicable for very large models
3) It is not possible to always seed correct number of faults to estimate reliability

3.4.4. Goel- okumoto model

$$\mu (t) = m (1 - \exp (-nt)) \quad (4)$$

m>0, n>0
m= initially taken total defect up to the time
n=rate where defects are detected
In this model we can estimate the number of errors found in the current program or the present model. This gives us the exact values of the errors. Blessy Thankachan[7] referred that the components of this model are:

1) Expected number of defects.
2) Roundness factor.
3) Test Time data.
4) Estimation Of model parameters.

As in this we clearly update the initial values where the errors were found and also the rate where it was detected it would be the easy way to calculate.
Assumptions:

1) The failures are exponentially distributed between execution times
2) The quantities which are available are constant up to some extent
3) The number of faults which will be detected in regular intervals are independent of each other

This model provides the correct estimation of errors in an effective way. It can correlate time with the environmental factors directly. If the risk factor is very high testing should be done continuously in this model. Sometimes it may give back the same errors if risk factor is high. Hence by using this method reliable software can be obtained in an efficient way.
Disadvantages:

1) Very time consuming
2) Costly method
3) Only possible if above assumptions are true
4) If the risk factor is high errors cannot be solved.

Calculation:

Table 1: Goel- Okumoto Model

M	n	T	μ(t)
5	2	2	14.08
15	5	2	37.7
25	10	2	70.4
30	15	2	98.1

Table 2: Goel- Okumoto Model

M	n	T	μ(t)
5	2	3	8.28
15	5	3	27.2
25	10	3	52.3
30	15	3	72.29

4. Proposed model

The main objective of any reliability model is to predict the behavior of the failure in the software while it is functioning. Now let us consider both mills model and also Goel-Okumoto model. The main disadvantage is based on the errors and also they cannot be applied for the large models. This is the primary drawback of both the models. The proposed model main objective is to overcome the disadvantage. So, in this we combine the models of Goel-Okumoto and Mill's error seeding method. By this method we can overcome the disadvantages of the method and used to determine the number of errors in an effective way. Let us consider the equations of Mill's model, i.e.

$$N= (n(r-k)/k)-1 \quad (5)$$

Here N indicates the total number of errors and consider Goel Okumoto model which also contains the term same as total number of errors but in a different notation. Let us consider them both as one term and can eliminate the other term. In Mill's model we will get less number of errors but there the time will not be included. So, we cannot know in how much time the errors will be recovered. Now let us consider Goel's model i.e.

$$\mu (t) = m (1 - \exp (-nt)) \quad (6)$$

In this we have time factor. So, if we combine this both the number of errors will be recovered and in the similar way time taken to done the process will also be known. As time plays an important Role if we detect less number of errors in more number of days which means rate of error occurrence is less. Now our model has both the factors that is time period and also number of errors occurred is also less.

$$N = (r - t) - 1/k (1 - \exp - nt) \quad (7)$$

N= total number of errors found after developing the model
T= time taken for the model to be performed
n= rate where errors are detected
k= errors which cannot be avoided
r= number of errors which were removed during debugging

Table 3: Hybrid Model

n	r	t	k	N
2	10	2	5	8.04
4	15	2	8	13.15
6	20	2	10	18.07
8	25	2	12	24.47

In the above set we have calculated for the time period of two days. Now let us consider by taking time in different ways.

Table 4:

n	r	T	k	N
2	10	3	4	3.47
5	25	3	12	11.64
10	36	3	19	21.78
15	48	3	27	29.85

In this model we have solved the primary drawback that is total number of errors found which means we are reducing the error rate using this model. The second disadvantage is that the both models are time taking. But using this method as there are less number of factors which can be observed and at the same time they can also be calculated. So, this is even used to overcome the other drawback. As we know time is a major component in any of the model its consumption should be as much less as possible. This model overcomes both the disadvantages in an efficient way. Hence here the number of errors detected are less. As we know reliability is a condition where the number of failures occurring should be less and its working should be high under certain conditions. Reliability is inversely proportional to the number of errors which means if there are more number of errors reliability will be less and vice- versa. So, this model has less number of errors compared to the above models. Therefore, this is achieving reliability.

5. Conclusion

By referring the above models we get a clear idea about different types of reliability models. Hence by implementing the above model we can reduce the error count as well as we are also improving the reliability of the system. Reliability includes the reduction of failure of the system or reduction in the occurrence of number of errors. If we implement the reliability factors in this model that improves the other factors and that can be used in the future for some models. This covers the disadvantage of both models and implements the better results. Therefore this model improves the reliability by reducing number of errors.

References

[1] Garima Chawlaet. Al., A Fault Analysis Based Model For Software Reliability Estimation, International Journal of Recent Technology And Engineering (Ijrte), Issn: 2277 -3878, Volume - 2, Issue - 3, July 2013.

[2] Felipe Febrero,Coral Calero,M. Ángeles Moraga,"Software Reliability Modeling Based On Iso/Iec Square",Journal Of Infor-mation And Software Technology,Volume 70 Issue C,Pages 18-29,February 2016 https://doi.org/10.1016/j.infsof.2015.09.006.

[3] Zhu, M., Zhang, X., Pham, H,"A comparison analysis of environmental factors affecting software reliability", Journal of Systems and Software, Volume 109 Issue C, Pages 150-160, November 2015. https://doi.org/10.1016/j.jss.2015.04.083.

[4] Nisha Sharma, Ms. Parveen Bano," A Survey of Software Reliability factor", Journal of Computer Engineering, Volume 12, Issue 1 PP 50-55 ,2013.

[5] Amanpreet Kaur, Kailash Bahl," Fault Seeding Models and Input Domain Based Models for Software Reliability Measurement and Improvement", International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 5, May 2016.

[6] Razeef Mohd., Mohsin Nazir,"Software Reliability Growth Models: Overview and Applications", Journal of Emerging Trends in Computing and Information Sciences, VOL. 3, NO. 9, SEP 2012.

[7] Blessy Thankachan, Pankaj Nagar," Application of Goel-Okumoto Model in Software Reliability Measurement", Special Issue of International Journal of Computer Applications on Issues and Challenges in Networking, Intelligence and Computing Tech-nologies, November 2012.