IMPROVED SOFTWARE QUALITY ASSURANCE TECHNIQUES USING SAFE GROWTH MODEL

M.Sangeetha,
Research Scholar,
Dept of CSE & IT
Coimbatore Institute of Technology,
Coimbatore.

K.M.SenthilKumar
Lecturer,
Kumaraguru College of Technology,
Dept of Mechatronics,
Coimbatore.

Dr.C.Arumugam
(S.G) Lecturer,
Dept of Mechanical Engineering,
Coimbatore Institute of Technology,
Coimbatore.

K. Akila
Lecturer,
Kumaraguru College of Technology,
Dept of Mechatronics,
Coimbatore.

***Abstract*—In our lives are governed by large, complex systems with increasingly complex software, and the safety, security, and reliability of these systems has become a major concern. As the software in today's systems grows larger, it has more defects, and these defects adversely affect the safety, security, and reliability of the systems. Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. Software divides into two pieces: internal and external quality characteristics. External quality characteristics are those parts of a product that face its users, where internal quality characteristics are those that do not.Quality is conformance to product requirements and should be free. This research concerns the role of software Quality. Software reliability is an important facet of software quality. It is the probability of failure-free operation of a computer program in a specified environment for a specified time. In software reliability modeling, the parameters of the model are typically estimated from the test data of the corresponding component. However, the widely used point estimators are subject to random variations in the data, resulting in uncertainties in these estimated parameters. This research describes a new approach to the problem of software testing. The approach is based on Bayesian graphical models and presents formal mechanisms for the logical structuring of the software testing problem, the probabilistic and statistical treatment of the uncertainties to be addressed, the test design and analysis process, and the incorporation and implication of test results. Once constructed, the models produced are dynamic representations of the software testing problem. It explains need of the common test-and-fix software quality strategy is no longer adequate, and characterizes the properties of the quality strategy.**

Keywords- Software reliability, uncertainty analysis, Bayesian Approach, MEP, MC, TTF.

1.INTRODUCTION

In software reliability modeling, the parameter of the model is typically estimated from the test data of the corresponding component. However, the widely used point estimators are subject to random variations in the data, resulting in uncertainties in these estimated parameters. Ignoring the parameter uncertainty can result in grossly underestimating the uncertainty in the total system reliability to apply the models for predicting the reliability of the component; the parameters of the models need to be known or estimated. Field data or data from components with similar functionalities are usually available to help estimate these parameters, but the estimators are subject to random variation because they are functions of random phenomena. Parameter uncertainty arises when the input parameters are unknown. Moreover, the reliability computed from the models, which are functions of these parameters, is not sufficiently precise when the parameters are uncertain.

2. SOFTWARE RELIABILITY AND ITS PROBLEM

Reliability engineering is an important aspect of many system development efforts and consequently there has been a great deal of research in the software based systems. One important activity included in reliability engineering is reliability prediction. Reliability is usually measured based on probability theory and, in general, mathematical statistics are used to estimate these probabilities. To control a complex industrial process, many heterogeneous components are needed to work together with highest reliability. The integration of heterogeneous components into mechatronics systems requires broadening the concepts and cooperation between different technical disciplines involved to develop a common conception of the future product and come up with an optimized solution. So the complexity of mechatronics system is to be broken by extracting the basic fundamental function of the different units. So, it is needed to abstract the reliability related peculiarities from each discipline and unify them in a way suitable for mechatronics as whole. The reliability definition stresses four elements namely i. probability ii. Intended functions iii. Time iv. Operating conditions.

The numerical evaluation of reliability is based on the total duration of failures or the frequency of failures [4]. So, most benefit from the use of reliability can be achieved at the early phases of the design of programmable mechatronics system to identify the adequate confidence about the systems and its components. The reliability analyses of various components involved in software based systems can be broadly classified in to

a. Software component reliability.
b. Interface and networking reliability.
c. Software controlled hardware reliability.

It has been noted that reliability of the large scaled electro-mechanical systems should be properly evaluated based on the subsystems reliability.

Definition for software "Set of programs, procedures and its related documentation for delivery to a customer. Software Quality deals with "fitness of Use" and "Satisfies user requirement".IEEE defines software quality as – a software feature or Characteristic used to assess the quality of a system [10].Software Quality is measured in different ways, using internal parameters, and external attributes [3] .Software reliability is Quality factor. Parameter definition is very important in Reliability. System level improved reliability factor is hard to find.

3. LITERATURE SURVEY

Reliability is frequently related to the probability of a failure occurring in the operational use of the system. Software systems are usually not considered to wear out with time. The only external factors that determine the output of a program are the input data. Faults in the software are essentially originated by human mistakes, such as a wrong specification, or instruction in the code. When a failure occurs, faults should be located & removed, so the system reliability tends to increase. Software reliability models are used to describe this process. They allow the estimation or prediction of the present or future reliability of a system.

Software reliability models consider different elements of the software project, such as the specification & codification of the programs, or information about the fault detection & correction process [2]. They are usually based on characteristics of the testing activity, and are classified according to the assumptions that each model does in its formulation. Some of them are based on time [3], [5], that is, they consider the time between failures. The parametric and traditional models assume in their analytic formulation a predetermined behavior where some parameters have to be adjusted, fitting the curve to the failure data. Their parameters are explicitly defined in the model, and have a physical interpretation. To adjust the parameters of the parametric models, there is always a set of assumptions that may not be suitable for most cases. The influence of external parameters and other peculiarities of a model can be eliminated if we have a model that is able to evolve itself based on the failure data collected during the initial test phase. Although they include parameters in their analytical formulation, they are also known as non-parametric reliability models because these parameters do not have a physical interpretation.

Software qualities well known models are McCall, Boehm, FURPS .The following sections will discuss briefly on models:

3.1. The McCall model (1977)

This model is constructed using tree-like fashion. This model holds quality factors usability which will be quantified [4].In this model Quality factors are not directly measured and set of metrics is needed to develop relationship.

3.2. The Boehm model (1978)

This Boehm model represents a hierarchical structure of characteristics, each of which contributes to total quality [4], [5]. Utility is broken in to different levels. This model is not considered for higher level.

3.3. The FURPS model (1987)

Hewlett-Packard developed a set of software quality factors that make up its name FURPS.It takes Functionality, Usability, Performance and supportability [6].
Disadvantage of this model is that it does not take into account the software product's portability [7].

3.6 The Systemic Quality Model (2003)

The systemic model is differed from the previous model mentioned above. This model is developed by identifying the relationship between product-process, efficiency-effectiveness and user-customer to obtain global systemic quality [8]. The model proposed focuses on product quality, which includes efficiency and effectiveness and the concept of systemic global quality. The disadvantages of this model are that it does not cover the user requirements and conformant aspects. Analysis done on the different models demonstrates that different quality characteristics associated with these different models.

4. QUALITY SAFE GROWTH MODEL

In semi-supervised learning, decision rule is to be learned from labeled and unlabeled data. In this framework, we motivate minimum entropy regularization, which enables to incorporate unlabeled data in the standard supervised learning. Our approach includes other approaches to the semi-supervised problem as particular or limiting cases. A series of experiments illustrates that the proposed solutions benefit from unlabeled data. Another characteristic challenge in software testing and reliability is the lack of available failure data from a single test, which often makes modeling difficult. This lack of data poses a bigger challenge in the uncertainty analysis of the software reliability modeling. The existing semi-supervised learning techniques are all not very effective for our case of lack of failure date.

4.1. Objective

The objective is to quantify the uncertainties in the software reliability model of system with correlated parameters. Challenges in software reliability is the lack of available failure data from a single test, which often makes modelling difficult. Using that data poses a bigger challenge in the uncertainty analysis of the software reliability modelling. For achieving good quality, we use reliability model using Bayes approach with TQM.

4. 2.RF Approach

RF (Reliability Factor), Reliability is the probability of a device performing its purpose adequately for the period intended under the given operating conditions. The definition brings into focus four important factors namely,

i. The reliability of a system is expressed as a probability [8].
ii. The system is required to give adequate performance [1].
iii. The duration of adequate performance is specified [3].
iv. The environmental or operating conditions are prescribed.

Reliability is usually measured in terms of probabilities. The theory of Bayesian statistics is a well established and the method has been applied in various areas including software, automation systems, medical diagnosis, geological explorations etc.

In the software based system, the uncertain variables are associated to each component where the uncertainty is expressed by probability density. The probability density expresses our belief or confidence in the various possible outcomes of variable. This probability depends conditionally on the status of other component based on different input.

The probabilities are estimated by means of statistical methods. Various statistical methods are available for estimating the reliability of the system, but these methods are not suitable for estimating the reliability of software based system, because, these methods have not considered the uncertainties of unknown parameters. The Bayesian statistical method considers the uncertainties of unknown parameters. So, Bayesian model is mainly used for estimating the reliability of software-based systems. This method is also used to predict the future of the system by the observed information.

Reliability is the probability that a system will operate without failure for a given time in a given environment. Note that reliability is defined for a given environment.

Since the test and operations environments are generally different, model results from test data may not apply to an operations environment. The "given time" in this definition may represent any number of actual data items, such as number of executions; number of lines of code traversed, or wall clock time.

The use of a model also requires careful definition of what a failure is. Reliability models can be run separately on each failure type and severity level. Models have been developed to measure, estimate and predict the reliability of computer software.

Software reliability has received much attention because reliability has always had obvious effects on highly visible aspects of software development: testing prior to delivery, and maintenance.

Early efforts focused on testing primarily because that is when the problems appeared. As technology has matured, root causes of incorrect and unreliable software have been identified earlier in the life cycle. This has been due in part to the availability of results from measurement research and/or application of reliability models.

Step 1. Take a prior factor p(a) with respect to a certain parameter, given a set of newly observed data.

Step 2. Using the newly observed data, estimate the parameter ‘f’.

Step 3. Derive $g=\int f'/4$ by using that formulae.

Step 4. Compare the values ‘f’ and ‘g’.

Step 4. If $(f < g)$ or $(f > g)$, the adjustment should be triggered. The estimated parameter ‘E’ is located at the extreme tails of the prior distribution.

Step 5. Adjust the degree factor d* for filtering and then recalculating the prior distribution and then repeat Step 1.

4.3. Measure Information

It can be used to approach physical systems from the point of view of information theory, because the probability distributions can be derived by avoiding the assumption that the observer has more information than is actually available. Information theory, particularly the definition of information in terms of probability distributions, provides a quantitative measure of ignorance (or uncertainty, or entropy) that can be maximized mathematically to find the probability distribution that is maximally unbiased

Entropy

If any of the probabilities is equal to 1 then all the other probabilities are 0 and we then know exactly which state the system is in. Since probabilities are used to cope with our lack of knowledge, and since one person may have more knowledge than another, it follows that two observers may, because of their different knowledge, use different probability distributions. In this sense probability, and all quantities that are based on probabilities, are subjective.

`Our uncertainty is expressed quantitatively by the information which we do not have about the state occupied. This information is

$$K=\sum_{i,j} p\ (T_{i+}T_j)\ \log_2\ (1/p\ (T_i)\ +j\ (T_i)) \qquad (1)$$

$$L=\sum_{i,j} (p\ (T_{i+}T_j)*g\ (T_{i+}T_j\ ) \qquad (2)$$

Information is measured in bits because we are using logarithms to base 2.

One person may have different knowledge of the system from another, and therefore would calculate a different numerical value for entropy. The Principle of Maximum Entropy is used to discover the probability distribution which leads to the highest value for this uncertainty, thereby assuring that no information is inadvertently assumed.

4.4. Reliability Model for All Modules of a System:

The system reliability can be evaluated using the architecture and relationship of the components.

The Markov model, SPN, fault tree analysis, and reliability block diagram are some popular tools for evaluating the system reliability, given the parameters of its contained components [8].

For example, a Markov chain is characterized by its state space, together with the transition Probabilities over time between these states. Usually, there are four steps to construct and solve the Markov chain models[7]:

1. Set up the Markov chain model.
2. List the Chapman-Kolmogorov equations.
3. Solve those equations to obtain state probabilities.
4. Obtain the reliability by summing up the probabilities of those reliable states.

→ Choose prior distributions based on previous knowledge either the results of earlier studies or non-scientific opinion.

Modern Parametric Bayesians and the normal model with unknown mean and variance

- Consider r and t^2 are unknown random variables.

$$Bi \sim n(r\ ,\ t^2) \qquad (3)$$

- we used the definition of conditional probability, so

$$p\ (r\ ,(IL+t^2) = p(r|(IL+t^2)p(IL+t^2)? \qquad (4)$$

- This is a conjugate distribution for the normal distribution with unknown mean and variance; the posterior distribution will also be normal-Inv-X^2.

In the previous sections, we have analyzed the parameter uncertainty of reliability model for one component based on Entropy and Bayes. Complicated software contains multiple modules. Many tools can be implemented to evaluate the system reliability, given the parameters of its contained components, such as the Markov models, Bayesian Network, Graph Theory, and Fault-Tree Analysis. Regardless of the tools used, the system reliability is a function combining the parameters of its components. As a result, the uncertainties in the parameters affect the whole system reliability. The purpose of this section is to study and quantify the uncertainty in the reliability of the complex system due to the uncertainty of the parameters in the numerous components of the system. Some general assumptions of the system-level analysis are listed as follows:

1. Consider system contains different modules.
2. Each module has its own reliability model and let j denote the set of parameters for the ith module's model, where j = 1; 2. .   . K.
3. For each module, the probability distribution of its model's parameters is known, which can be derived using formula2 and 4.
4. The failures of different components are'd' independent, the system reliability can be Calculated by the function of modules Parameters as $d = f\ (v_1,\ v_2,\ldots\ldots,\ v_j)$.

Based on the above assumptions, an MC simulation is presented for generally analyzing the uncertain system reliability. It is difficult to use analytic methods for combining the distributions of numerous parameters to derive the probability density function of the system reliability, especially for complicated systems with complex architecture and many components. Hence, the MC simulation becomes a practical way to make the uncertainty analysis of the complicated system tractable. Algorithm 1 provides a general MC approach for the uncertainty analysis in a complicated system.

4.5. INFORMATION VALIDATION

The posterior distributions validate the posterior of modules by using the following method:

- Calculate the mean values of the posterior distributions and obtain parameters.
- Using, these parameters can be applied to the test.
- Compare the predicted TimeToFailures with the real observed TimeToFailures, by calculating the mean square error.

  M (t) =b(1-exp(-¢(t)), a>0 (5)
- If the mean square error m(t)>d(Threshold value)is then the model does not fit the observed data.
- The adjustment (such as trying another model or further filtering the subjective information) should be applied.

4.6. SAFE GROWTH MODEL(SGRM)

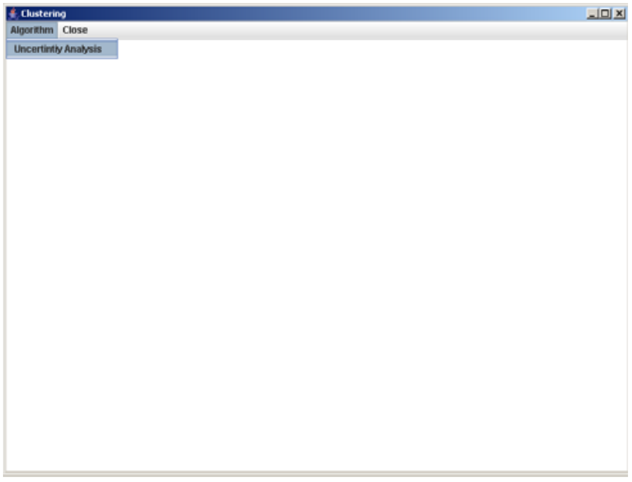
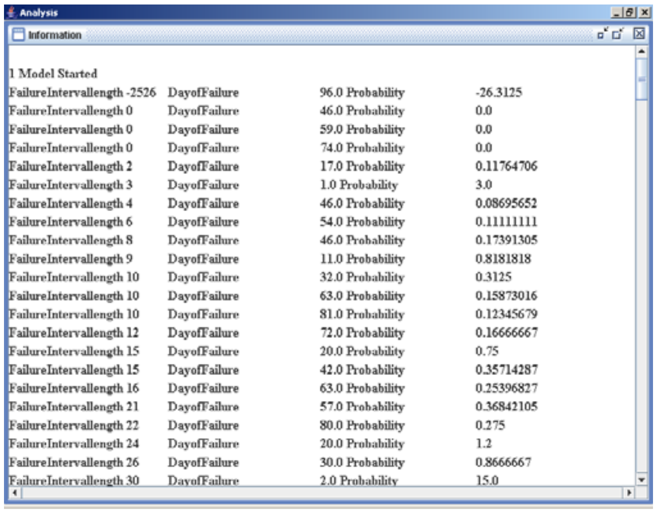
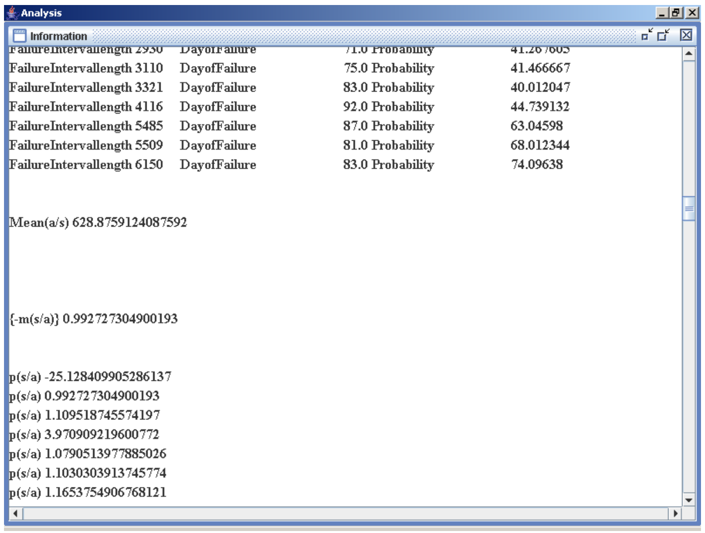


Figure 1: Main Screen



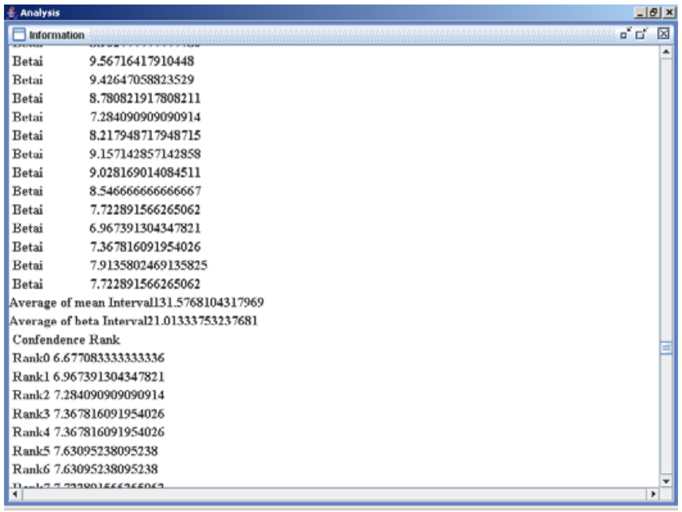
```
1 Model Started
FailureIntervallength -2526   DayofFailure    96.0 Probability    -26.3125
FailureIntervallength 0       DayofFailure    46.0 Probability    0.0
FailureIntervallength 0       DayofFailure    59.0 Probability    0.0
FailureIntervallength 0       DayofFailure    74.0 Probability    0.0
FailureIntervallength 2       DayofFailure    17.0 Probability    0.11764706
FailureIntervallength 3       DayofFailure    1.0 Probability     3.0
FailureIntervallength 4       DayofFailure    46.0 Probability    0.08695652
FailureIntervallength 6       DayofFailure    54.0 Probability    0.11111111
FailureIntervallength 8       DayofFailure    46.0 Probability    0.17391305
FailureIntervallength 9       DayofFailure    11.0 Probability    0.8181818
FailureIntervallength 10      DayofFailure    32.0 Probability    0.3125
FailureIntervallength 10      DayofFailure    63.0 Probability    0.15873016
FailureIntervallength 10      DayofFailure    81.0 Probability    0.12345679
FailureIntervallength 12      DayofFailure    72.0 Probability    0.16666667
FailureIntervallength 15      DayofFailure    20.0 Probability    0.75
FailureIntervallength 15      DayofFailure    42.0 Probability    0.35714287
FailureIntervallength 16      DayofFailure    63.0 Probability    0.25396827
FailureIntervallength 21      DayofFailure    57.0 Probability    0.36842105
FailureIntervallength 22      DayofFailure    80.0 Probability    0.275
FailureIntervallength 24      DayofFailure    20.0 Probability    1.2
FailureIntervallength 26      DayofFailure    30.0 Probability    0.8666667
FailureIntervallength 30      DayofFailure    2.0 Probability     15.0
```

Figure 2: Calculating Failure Interval length



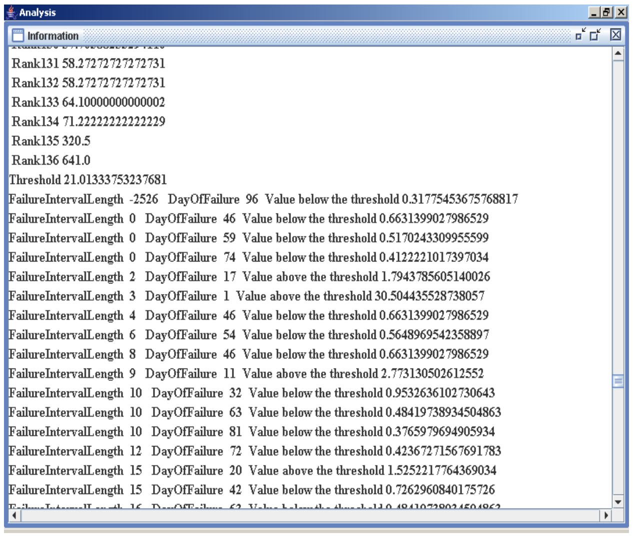
```
FailureIntervallength 3110    DayofFailure    75.0 Probability    41.466667
FailureIntervallength 3321    DayofFailure    83.0 Probability    40.012047
FailureIntervallength 4116    DayofFailure    92.0 Probability    44.739132
FailureIntervallength 5485    DayofFailure    87.0 Probability    63.04598
FailureIntervallength 5509    DayofFailure    81.0 Probability    68.012344
FailureIntervallength 6150    DayofFailure    83.0 Probability    74.09638


Mean(a/s) 628.8759124087592



{-m(s/a)} 0.992727304900193


p(s/a) -25.128409905286137
p(s/a) 0.992727304900193
p(s/a) 1.109518745574197
p(s/a) 3.970909219600772
p(s/a) 1.0790513977885026
p(s/a) 1.1030303913745774
p(s/a) 1.1653754906768121
```

Figure 3: calculating the mean value of failure data.



```
Betai     9.56716417910448
Betai     9.426470588235294
Betai     8.780821917808211
Betai     7.284090909090914
Betai     8.217948717948715
Betai     9.157142857142858
Betai     9.028169014084511
Betai     8.546666666666667
Betai     7.722891566265062
Betai     6.967391304347821
Betai     7.367816091954026
Betai     7.9135802469135825
Betai     7.722891566265062
Average of mean Interval131.5768104317969
Average of beta Interval21.01333753237681
Confendence Rank
Rank0 6.677083333333336
Rank1 6.967391304347821
Rank2 7.284090909090914
Rank3 7.367816091954026
Rank4 7.367816091954026
Rank5 7.63095238095238
Rank6 7.63095238095238
```

Figure 4: Calculating the Average mean interval and Confidence Rank.

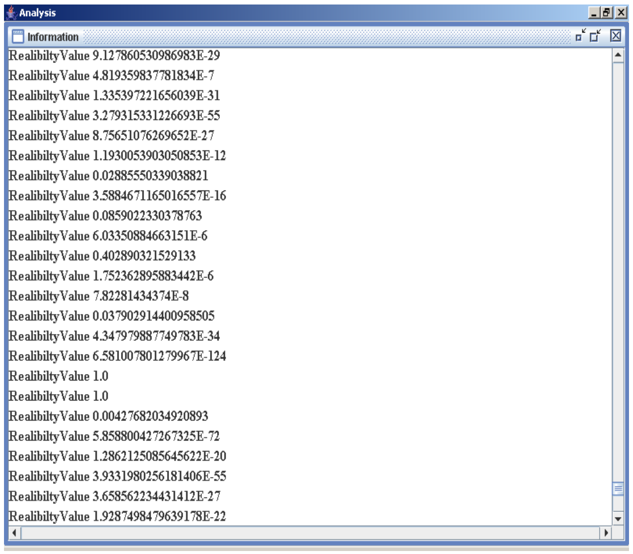


```
Rank131 58.2727272727272731
Rank132 58.2727272727272731
Rank133 64.10000000000002
Rank134 71.22222222222229
Rank135 320.5
Rank136 641.0
Threshold 21.01333753237681
FailureIntervalLength  -2526   DayOfFailure  96  Value below the threshold 0.3177545367576881 7
FailureIntervalLength  0   DayOfFailure  46  Value below the threshold 0.6631399027986529
FailureIntervalLength  0   DayOfFailure  59  Value below the threshold 0.5170243309955599
FailureIntervalLength  0   DayOfFailure  74  Value below the threshold 0.4122221017397034
FailureIntervalLength  2   DayOfFailure  17  Value above the threshold 1.7943785605140026
FailureIntervalLength  3   DayOfFailure  1  Value above the threshold 30.504435528738057
FailureIntervalLength  4   DayOfFailure  46  Value below the threshold 0.6631399027986529
FailureIntervalLength  6   DayOfFailure  54  Value below the threshold 0.5648969542358897
FailureIntervalLength  8   DayOfFailure  46  Value below the threshold 0.6631399027986529
FailureIntervalLength  9   DayOfFailure  11  Value above the threshold 2.773130502612552
FailureIntervalLength  10   DayOfFailure  32  Value below the threshold 0.9532636102730643
FailureIntervalLength  10   DayOfFailure  63  Value below the threshold 0.48419738934504863
FailureIntervalLength  10   DayOfFailure  81  Value below the threshold 0.3765979694905934
FailureIntervalLength  12   DayOfFailure  72  Value below the threshold 0.42367271567691783
FailureIntervalLength  15   DayOfFailure  20  Value above the threshold 1.5252217764369034
FailureIntervalLength  15   DayOfFailure  42  Value below the threshold 0.726296084017572 6
```

Figure 5: Calculating the Threshold Value

Model		
Failure Number	Failure Interval Length	Day of Failure
1	5	1
2	73	1
3	141	1
4	491	5
5	5	5
6	5	5
7	28	5
8	138	5
9	478	9
10	325	9

Table 1: Frequency table



```
RealibiltyValue 9.127860530986983E-29
RealibiltyValue 4.819359837781834E-7
RealibiltyValue 1.335397221656039E-31
RealibiltyValue 3.279315331226693E-55
RealibiltyValue 8.75651076269652E-27
RealibiltyValue 1.1930053903050853E-12
RealibiltyValue 0.02885550339038821
RealibiltyValue 3.588467116501655 7E-16
RealibiltyValue 0.085902233037876 3
RealibiltyValue 6.03350884663151E-6
RealibiltyValue 0.402890321529133
RealibiltyValue 1.752362895883442E-6
RealibiltyValue 7.82281434374E-8
RealibiltyValue 0.03790291440095850 5
RealibiltyValue 4.347979887749783E-34
RealibiltyValue 6.581007801279967E-124
RealibiltyValue 1.0
RealibiltyValue 1.0
RealibiltyValue 0.004276820349208 93
RealibiltyValue 5.858800427267325E-72
RealibiltyValue 1.2862125085645622E-20
RealibiltyValue 3.9331980256181406E-55
RealibiltyValue 3.658562234431412E-27
RealibiltyValue 1.9287498479639178E-22
```

Figure 6: the Reliability Value of Model

Safe Growth Model Quality Values:

k	L	p(r ,(IL+t^2)	m(t)	Rank	RealibiltyValue
0.981309	0	0	662.0769	242.4507	0.073452884
23.55142	0	0	391.2273	245.9143	0.936789997
1.358736	0	0	358.625	296.7931	0.911564803
1.59917	12.5	12.5	17214	344.28	0.911564803
3.140189	0.5	0.5	3442.8	344.28	0.595401972
6.476639	0.073964	0.073964	662.0769	358.625	0.677649082
2.564065	0.198217	0.198217	637.5556	358.625	1.43E-04
6.476639	2.42	2.42	1721.4	358.625	0.902433272
72.61686	15.68	15.68	3442.8	358.625	0.665728529
24.94034	1.300728	1.300728	555.2903	358.625	0.825561533

Table 2: Reliability Model Values

5. PERFORMANCE MEASURES

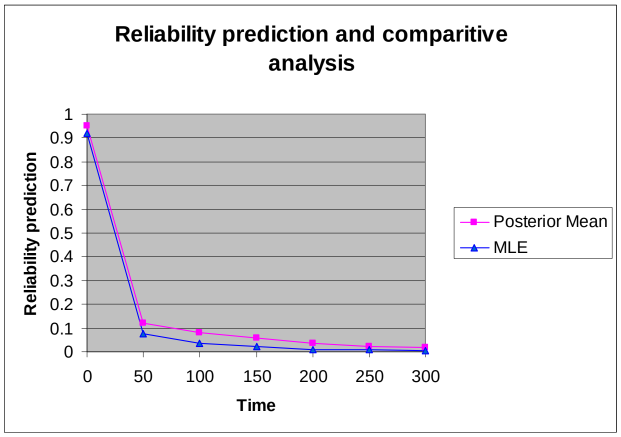


Figure 7: Reliability prediction and comparative Analysis.

In general, the reliability of the system decreases as time increases but the reliability of posterior mean always lies above the mean of MLE. Both point estimate methods of MLE and posterior mean can predict a close reliability trend, so the new method using the posterior mean can be an alternative way for the point estimate of the parameters. More importantly, using the posterior probability distribution, we can further analyze the uncertainty of the predicted reliability. In addition, the confidence intervals depend on both the prior distribution and the new observations.

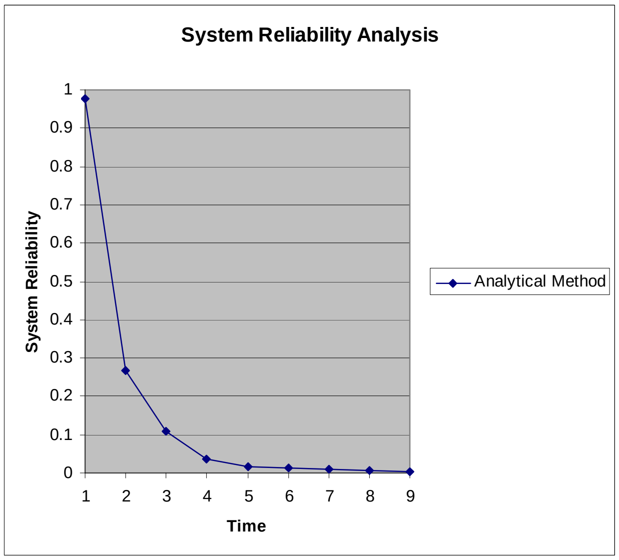


Figure 8 System Reliability Analysis

From the analytical results of uncertainty analysis in Fig. 8, we find that, during the initial period of the reliability prediction, the system reliability is high, indicating that the uncertainty of the software reliability is low. Then, the confidence interval increases and reaches the maximum around the middle part. At the latter part, the mean value of system reliability is also small so that the comparative uncertainty is still large.

6. CONCLUSIONS

This proposed work gives the solution for uncertainty problems in reliability modeling on system level. This safe growth model solves the challenges for the dearth of data by using quality factors. From the similar models, we have taken expert knowledge, historical data, and developmental environments .This expert knowledge involved in analyzing the uncertainty and for compensating insufficient failure data. After analyzing the problem, this work further extends to more complicated systems that contain numerous components, each with its own respective distributions and uncertain parameters.

REFERENCES

[1] G.L. Eyink and S. Kim, “A Maximum Entropy Method for Particle Filtering,” J. Statistical Physics, vol. 123, no. 5, pp. 1071-1128, 2005.
[2] A.L. Goel and K. Okumoto, “Time Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures,” IEEE Trans. Reliability, vol. 28, pp. 206-211, 1979.
[3] W. Suryn, A. Abran, P. Bourque & C. Laporte, “Software product quality practices: Quality measurement and evaluation using TL9000 and ISO/IEC9126,” *Proceeding of the 10th International Workshop, Software Technology and Engineering Practice (STEP)* 2002.
[4] F.E. Norman & S.L. Pfleeger, *Software Metrics: ARigorous and Practical Approach,* Second Edition. Boston: PWS Publishing, 1997.
[5] K. Khosravi, & Y.G. Gueheneuc, “ A quality model for design patterns”,2004.
[6] M. Goldstein, “Subjective Bayesian Analysis: Principles and Practice,” Bayesian Analysis, vol. 1, no. 3, pp. 403- 420, 2006.
[7] B.R. Haverkort and A.M.H. Meeuwissen, “Sensitivity and Uncertainty Analysis of Markov-Reward Models,” IEEE Trans. Reliability, vol. 44, no. 1, pp. 147-154, 1995
[8] D.E. Holmes, “Toward a Generalized Bayesian Network,” Proc. Am. Inst. Physics Conf.—Bayesian Inference and Maximum Entropy Methods in Science and Eng., vol. 872, pp. 195-202, 2006.
[9] M. Ortega, M. Perez, & T. Rojas, “Construction of systemic quality model for evaluating a software product”, *Software Quality Journal* 11: 219-242, 2003.
[10] E.T. Jaynes, “Information Theory and Statistical Mechanics,” Statistical Physics, pp. 181-218, 1963.
[11] C.Y. Tseng, “Entropic Criterion for Model Selection,” Physica A: Statistical and Theoretical Physics, vol. 370, no. 2, pp. 530-538, 2005.
[12] K.S. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Applications. Prentice-Hall, 1982.
[13]D.A. Wooff, M. Goldstein, and F.P.A.Coolen,“Bayesian Graphical Models for Software Testing,” IEEE Trans. Software Eng., vol. 28, no. 5, pp. 510-525, May 2002.
[14] IEEE. “IEEE standard for a software quality Metrics Methodology”, 1993. [August 20, 2005].
[15] M. Xie, Y.S. Dai, and K.L. Poh, Computing System Reliability: Models and Analysis. Kluwer Academic, 2004.
[16] Liao H,Enke D., and Wiebe H.,”An Expert Advisory Systems for ISO 9001 Quality System”,Expert Systems with Applications,Vol 27,pp.313-322,2004.
[17] J. Musa, A. Iannino, and K. Okumoto, *Software engineering and managing software with reliability measures*. : McGraw-Hill, 1987.
[18] C.-Y. Huang and C.-T. Lin, “Software reliability analysis by considering fault dependency and debugging time lag,” *IEEE Trans. Reliability*, vol. 53, no. 3, pp. 436–45 0, 2006.
[19] P. Moranda, “Predictions of software reliability during debugging,” in *Proceedings of the Annual Reliability Maintainability Symposium*, 1975.
[20] P. Moranda and Z. Jelinski, Final Report on Software Reliability Study McDonnall Douglas Astronautics Company, 1972, Tech. Rep.. [5] J. Musa, “A theory of software reliability and its application,” *IEEE Trans. Software Engineering*, pp. 312–327, 1975.
[21] N. Karunanithi, D. Whitley, and Y. K. Malaiya, “Prediction of software reliability using connectionist models,” *IEEE Trans. Software Engineering*, vol. 18, no. 7, pp. 563–574, July 1992.
[22] G. A. Souza and S. R. Vergilio, “Modeling software reliability growth with artificial neural networks,” in *IEEE Latin American Test Workshop*, Buenos Aires, Argentina, March 2006, pp. 165–170.
[23] J. Koza, *Genetic programming: On the programming of computers by means of natural selection*. : MIT Press, 1992. [9] J. Holland, *Adaptation in natural and artificial systems*. : MIT Press, 1975.
[24] E. O. Costa, S. R. Vergilio, A. Pozo, and G. A. Souza, “Modeling software reliability growth with genetic programming,” in XVI International Symposium of Software Reliability Engineering, USA, November 2005, IEEE Computer Society.
[25] G. Paris, D. Robiliard, and C. Fonlupt, “Applying boosting techniques to genetic programming,” in IEEE International Joint Conference on Neural Networks, 2004, pp. 1163–1168.
[26] D. Solomatine and D. Shrestha, “Adaboost-rt: A boosting algorithm for regression problems,” Intelligent Artificial Evolution, pp. 312–326, 2001.

M.Sangeetha received her M.Tech in Information Technology and her M.B.A in General.She is pursuing Ph.D in Computer Science Engineering. Currently, she is a Lecturer of Computer Science Engineering and Information Technology Department, Coimbatore Institute of Technology, Coimbatore.Her area of interests includes, Software Quality, Software Testing, Software Requirement Specification, Resource Management Technique and Information Security.