Volume 5, Issue 2, February 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Comparative Analysis of Reliability Models – Based on Uncertainty Factors

Arashdeep Kaur
CSE, Punjab Technical University
Punjab, India

Abstract— Quality, Schedule and cost are considered as important characteristics of software product. Reliability and availability are the measures considered for the software quality. Number of failures and Time between failures is considered as important uncertainty factors in the software reliability. But still many limitations are there in the recent approaches that do not consider these factors as an influential factor in terms of software reliability. CASRE tool have been used to measure the reliability of software using some of the models.

Index Terms-Reliability Analysis, Quality, Failure Rate, SRGM's.

I. INTRODUCTION

Uncertainty is the deviation of the reliability estimate given by the model, from the 'true' reliability of the system. Factors influencing uncertainty include characteristics of software, such as program complexity, test coverage, development environment and many others, appearing during the development life cycle. The uncertainty factors considered for this study are:

- Number of Failures
- Time Between Failures

Since software does not deprecate like hardware, the reliability of software stays constant over time if no changes are made to the code or to the environmental conditions including the user behaviour. However, if each time after a failure has been experienced the underlying fault is detected and perfectly fixed, and then the reliability of software will increase with time [17]. Typically this is the situation during the testing phase(s). As each failure occurrence initiates the removal of a fault, the number of failures that have been experienced by time t, denoted by M (t), can be regarded as a reflected image of reliability growth. Models trying to explain and describe the behaviour of M (t) are called software reliability growth models (SRGMs). Each SRGM implicitly assumes a certain functional form of M (t). Using the failure time's t1, t2 ... or the times between failures Δt1, Δt2 ... collected during a testing project, the parameters of a SRGM can be estimated. The mean value function μ (t) of a SRGM represents the expected number of failures experienced by time t according to the model. The derivative of the mean value function with respect to time, dμ(t) / dt , is named failure intensity λ(t). It represents the limit of the expected number of failures experienced in time interval [t; t+Δt)] for Δt approaching zero.

The failure intensity should not be mistaken for the hazard rate of the application, z (Δt | ti−1), the probability density for the i^{th} failure being experienced at t_{i-1} + Δt conditional that the $(i-1)^{st}$ failure has occurred at t_{i-1}. In the field of survival analysis the hazard rate of a component is often denoted by λ (t). Moreover, some researchers in software reliability call the program hazard rate "failure rate r (t)" or even "failure intensity λ (t)". The fact that the program hazard rate z $(\Delta t \mid t_{i-1})$ is indeed equal to the failure intensity at time t_{i-1} +Δ t for software reliability growth models of the Poisson type may contribute to this confusion. Therefore the nature of the failure process depends on the software product. We need to study the failure behaviour of software that further affects the software reliability. Research is still going on these uncertainty factors considering both the black box and white box models. In this research, main aim is to carry on with the black box models together with the factors to analyse the reliability issues [1].

II. SOFTWARE RELIABILITY GROWTH MODELS

Although some historical SRGMs have been widely adopted to predict software reliability, researchers believe they can further improve the prediction accuracy of these models by adding other important factors which affect the final software quality.

The software reliability models are broadly classified into black box (single system) and white box (multi component software) models. The black box models are classified further into models based on Inter failure times, Failure count and static models. Markov [10] assumptions are also made for some models and they are overlapping with the failure count and inter failure time modelling. Note the fact that all known models can be extended to be a Bayesian model. If the model is Bayesian then we are estimating the model parameters using Bayesian techniques. A number of stochastic models have been developed and tested against observed software system failure data. A small number of models are

being used to monitor the reliability performance of software systems as they progress through the various phases of the software life cycle [3]. To estimate the software failure rate from the observed failures, the first approach utilizes software reliability growth models (SRGM) in later stages of software development cycle. The number of failures that have been experienced by time t, denoted by *M* (*t*), can be regarded as a reflected image of reliability growth. Each SRGM implicitly assumes a certain functional form of mean value function *M* (*t*). Using the number of failures or the times between failures as the uncertainty factors considered during a testing project, the parameters of a SRGM like expected number of failures by the time *t* or failure intensity can be estimated [16].

A broad classification of SRGMs is given in Table 2.1. SRGMs are classified under three major groups: finite, infinite based on the total number of failures expressed in infinite time and Bayesian models. These models describe how observation of failures and correcting the underlying faults affect the reliability of software. These models are applicable also to assessing the reliability of software in operational use, when the latest reliability estimate given by the model is used [17].

Table 1 Classification of SRGM's

Finite		**Infinite**	**Bayesian Model**
Exponential	**Weibull & Gamma**	**Exponential**	
Jelinski-Moranda	Weibull	Geometric	Littlewood-Verrall
Musa Basic Execution Time	S-Shaped Reliability Growth	Musa-Okumoto Logarithmic	
Goel-Okumoto		Duane	
Schneidewind			

III. METHODOLOGY USED

From among the various categories of SRGM's, four models were considered for the software reliability estimation. They are:

- Yamada S-Shaped Model
- Schneidewind Model
- Littlewood Verrall Model [4]
- Musa Basic Execution Time Model [11]

For the above two models, 'Number of failures' uncertainty factor was taken into account for reliability estimation. For the next two models influence of 'Time between failures' uncertainty factor was analyzed. There are two methods of parameter estimation which are known as maximum-likelihood (ML) and least-squares (LS). The default mechanism is ML. However, ML may not always produce parameter estimates, and takes more computation time than LS. LS take less computation time to estimate parameters, and always find a parameter estimate. Various models are there for reliability estimation, each one of them takes some assumptions. No software is totally perfect. They have some constraints or hidden factors in them. But the software should be made reliable enough to satisfy the customer according to their needs. In our work we have gone through with few uncertainty factors using these models to analyse the influence of these factors on the reliability of software. Various phases followed in this research process are as under:

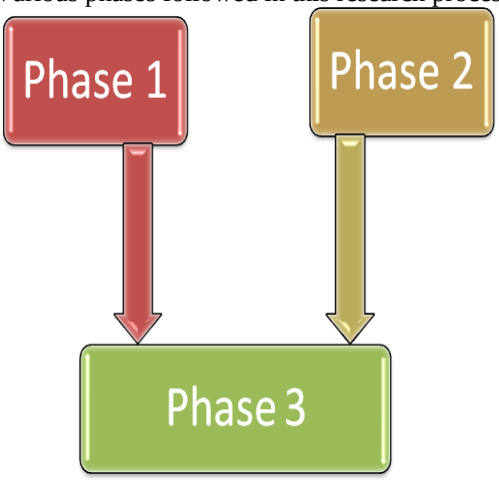


Fig 1 Methodology used for the comparative analysis of models.

PHASE 1- Reliability estimation of S-Shaped [7] and Schneidewind model [8] using *number of failures* as uncertainty factor.

Step 1- Generation of the dataset in the required format acceptable by the tool.

Step 2- Based on the above dataset for various test intervals, the number of failures encountered is being analyzed.

Step 3- For the same number of failures generated during step 2, reliability estimation for Schneidewind and S-shaped model is done.

PHASE 2- Reliability estimation of Musa Basic Execution time [5] and Littlewood Verrall model [4] using *Time Between Failures* as uncertainty factor.

Step 1- Generation of the dataset in the required format acceptable by the tool.

Step 2- Based on the above dataset, Time between Failure factors is applied. Many filters from the tool were used to have accurate details. The graph for the same is shown in the result (v) section.

Step 3- For the same Uncertainty Factor considered during step 2, reliability estimation for Musa Basic and Littlewood Verrall model was done.

PHASE 3- Comparative analysis of SRGM's was done for both the models. Each factor has different influence on the reliability of the software.

IV. TOOL USED – CASRE 3.0

CASRE (Computer Aided Software Reliability Estimation) [6] was developed as a software reliability measurement tool that is easier for non-specialists in software reliability engineering to use than many other currently-available tools. Although, some new features regarding the SRGM models are required to be added in CASRE. It incorporates the mathematical modeling capabilities of the public domain tool SMERFS (Statistical Modeling and Estimation of Reliability Functions for Software), and runs in a Microsoft Windows environment. CASRE was initially designed and implemented in a Windows environment as a stand-alone tool. Afterwards, it was subsequently ported to the UNIX operating system, using Tcl/Tk [9] to re-implement its user interface. The user interface of CASRE is menu driven. Users are guided through the selection of a set of failure data and the execution of a reliability model by selectively enabling pull-down menu options.

Modeling Results can be displayed using Graphical representation. After one or more models have been executed, the modeling results and comparisons are drawn in a graphics display window. Many options are available to display different kind of results. Users manipulate this window's controls to display the results in a variety of ways. Users may also display the results in a tabular fashion if they wish. CASRE's problem solvers for reliability models are implemented in FORTRAN and C.

The mandatory factor that is required in this behalf is the data set which is made based on the uncertainty factors. Version 3.0 includes two trend tests that may be used to determine whether it is even appropriate to apply a software reliability model to a set of failure data. These two tests, the running arithmetic average and the Laplace test, allow users to determine whether a set of failure data indicates that the system's reliability is increasing during test, whether it is decreasing, or whether there is no discernible trend. If the failure data shows decreasing reliability or no discernible trend, applying reliability models to the data is not indicated. Version 3.0 also has a simpler input format than previous versions - for input data entered as the number of failures in a test interval of a specified length, eliminating a rarely-used model further help the removal of three fields specific to that model. However, version 3.0 will still read input data formatted for previous versions. Previous versions of CASRE [6] would let users select the model, and then force them to select one of the variants - for example, if a user wanted to run the Schneidewind model, they would first select that model, and then select one of the three variants. This made the selection mechanism more complicated, and allowed users to run only one model variant at a time. Version 3.0 treats each model variant as a separate model - users now simply select all of a model's variants they want to run. This makes the selection mechanism somewhat simpler and more consistent, and users are now able to simultaneously run all variants of a model.

A. Applicability

CASRE can be applied starting after unit test and continuing through system test, acceptance test, and operations. You should only apply CASRE to modules for which you expect to see at least 40 or 50 failures. If you expect to see fewer failures, you may reduce the accuracy of your estimates. Experience shows that at the start of software test, modules having more than about 2000 source lines of executable code will tend to have enough faults to produce at least 40 to 50 failures.

B. Dataset Used

The data used is failure data about particular software available on "DACS Software Reliability" datasets [2]. Failure data files used by CASRE must be in the format acceptable by tool. One way to create these files is to use a word processor or text editor to create ASCII text files of the appropriate format. Text editors, word processors, and other applications can be invoked as stand-alone applications. If you have a database in which information about failures is tracked during testing, however, you can use whatever capabilities your database program has to create a delimited ASCII file having the format.

V. RESULTS

A. For S-Shaped Model – Generated graph for Number of Failures factor using hann filter.

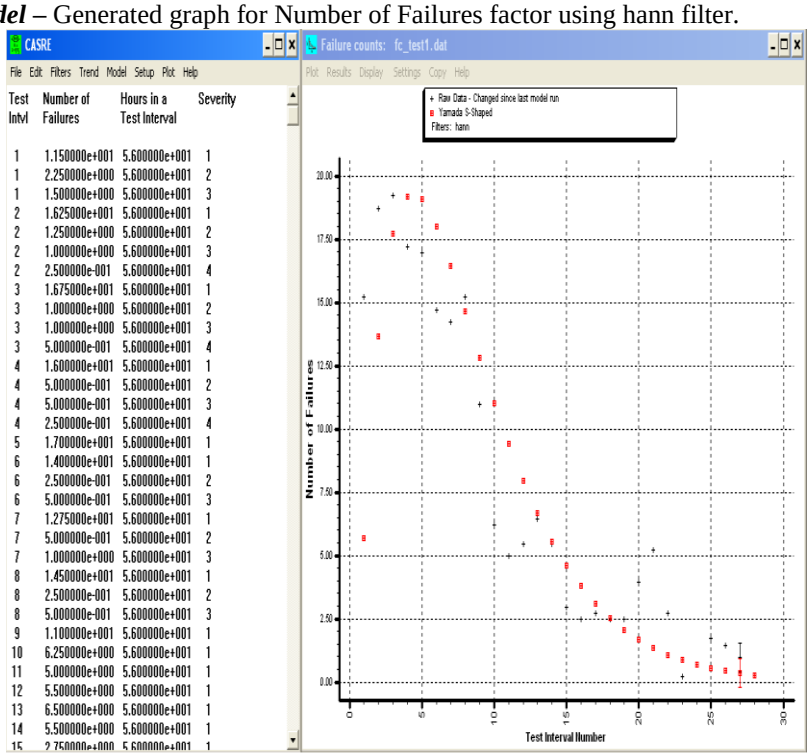


Fig 2 Snapshot of Number of Failures graph using Yamada s-shaped model

Analysis of S-Shaped model:

- Assumption: Total number of faults is finite.
- Graph reveals the variation of number of faults detected with respect to testing time interval. During initial phase of testing time the faults detected are high for few intervals then starts decreasing.

Reliability for S-Shaped model: Variation in the reliability curve for some hour's with respect to interval number.

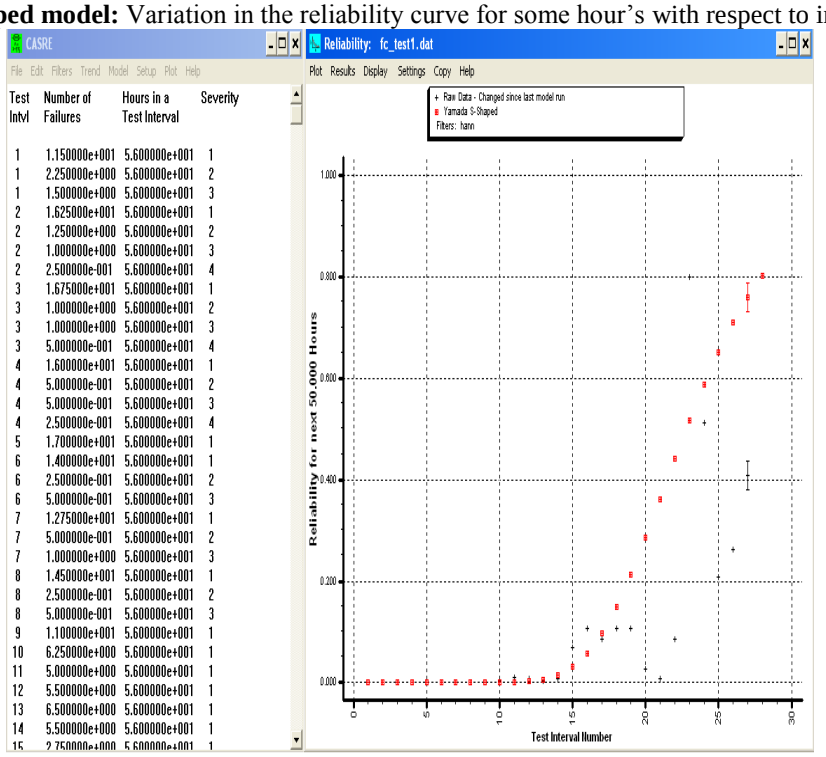
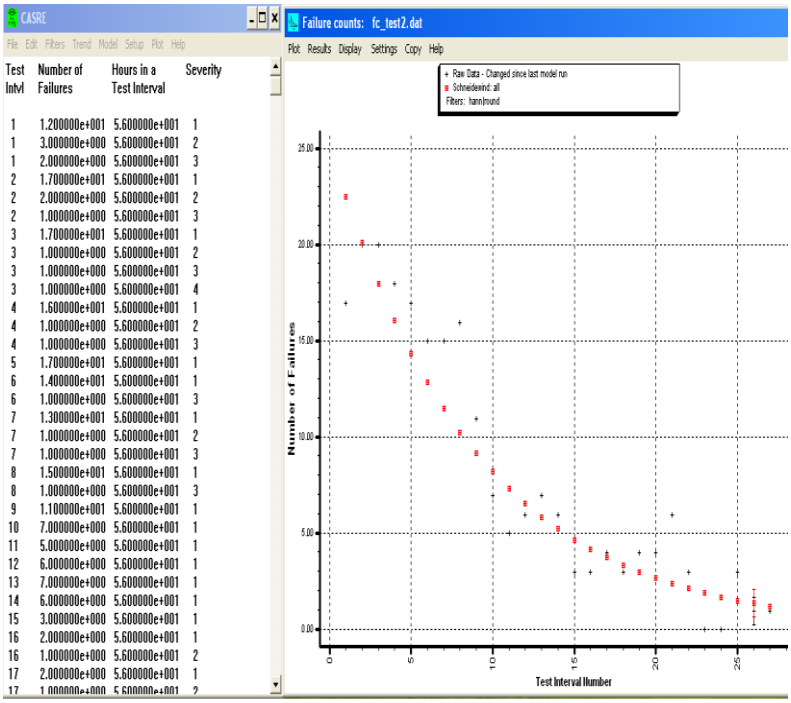


Fig 3 Snapshot of Reliability of S-shaped models

Analysis: In case of Yamada S-shaped model the reliability curve for number of failures data shows some variation as the time interval number increases. At the start point the reliability curve is constant for first four equal length time intervals. After the first four intervals, reliability of model increases with increase in the interval number. So the reliability of the model has strong relation with the test interval number.

B. FOR SCHNEIDEWIND MODEL – GENERATED GRAPH FOR NUMBER OF FAILURES USING TEST INTERVAL LENGTH IN HOURS.



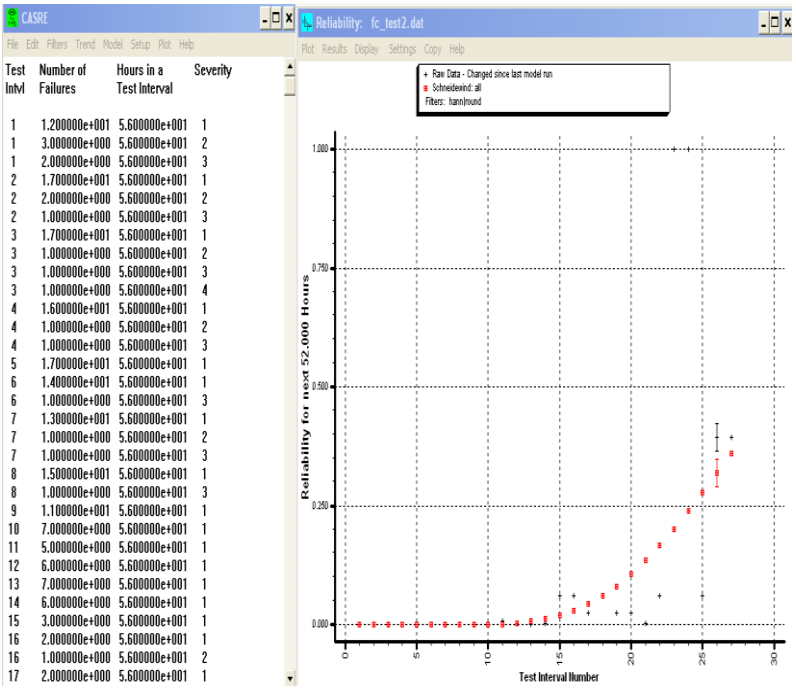
Test Intvl	Number of Failures	Hours in a Test Interval	Severity
1	1.200000e+001	5.600000e+001	1
1	3.000000e+000	5.600000e+001	2
1	2.000000e+000	5.600000e+001	3
2	1.700000e+001	5.600000e+001	1
2	2.000000e+000	5.600000e+001	2
2	1.000000e+000	5.600000e+001	3
3	1.700000e+001	5.600000e+001	1
3	1.000000e+000	5.600000e+001	2
3	1.000000e+000	5.600000e+001	3
3	1.000000e+000	5.600000e+001	4
4	1.600000e+001	5.600000e+001	1
4	1.000000e+000	5.600000e+001	2
4	1.000000e+000	5.600000e+001	3
5	1.700000e+001	5.600000e+001	1
6	1.400000e+001	5.600000e+001	1
6	1.000000e+000	5.600000e+001	3
7	1.300000e+001	5.600000e+001	1
7	1.000000e+000	5.600000e+001	2
7	1.000000e+000	5.600000e+001	3
8	1.500000e+001	5.600000e+001	1
8	1.000000e+000	5.600000e+001	3
9	1.100000e+001	5.600000e+001	1
10	7.000000e+000	5.600000e+001	1
11	5.000000e+000	5.600000e+001	1
12	6.000000e+000	5.600000e+001	1
13	7.000000e+000	5.600000e+001	1
14	6.000000e+000	5.600000e+001	1
15	3.000000e+000	5.600000e+001	1
16	2.000000e+000	5.600000e+001	1
16	1.000000e+000	5.600000e+001	2
17	2.000000e+000	5.600000e+001	1

Fig 4 Snapshot of Number of Failures Graph using Schneidewind model

Analysis of Schneidewind model:

- Assumption: As faults are corrected no new faults are introduced and are finite.
- If same test interval is executed the second time then the number of faults are reduced. So the analysis concludes that failures decreases as the time interval number increases.

Reliability of Schneidewind model: variation of reliability of model with respect to test interval number on number of failure information



Test Intvl	Number of Failures	Hours in a Test Interval	Severity
1	1.200000e+001	5.600000e+001	1
1	3.000000e+000	5.600000e+001	2
1	2.000000e+000	5.600000e+001	3
2	1.700000e+001	5.600000e+001	1
2	2.000000e+000	5.600000e+001	2
2	1.000000e+000	5.600000e+001	3
3	1.700000e+001	5.600000e+001	1
3	1.000000e+000	5.600000e+001	2
3	1.000000e+000	5.600000e+001	3
3	1.000000e+000	5.600000e+001	4
4	1.600000e+001	5.600000e+001	1
4	1.000000e+000	5.600000e+001	2
4	1.000000e+000	5.600000e+001	3
5	1.700000e+001	5.600000e+001	1
6	1.400000e+001	5.600000e+001	1
6	1.000000e+000	5.600000e+001	3
7	1.300000e+001	5.600000e+001	1
7	1.000000e+000	5.600000e+001	2
7	1.000000e+000	5.600000e+001	3
8	1.500000e+001	5.600000e+001	1
8	1.000000e+000	5.600000e+001	3
9	1.100000e+001	5.600000e+001	1
10	7.000000e+000	5.600000e+001	1
11	5.000000e+000	5.600000e+001	1
12	6.000000e+000	5.600000e+001	1
13	7.000000e+000	5.600000e+001	1
14	6.000000e+000	5.600000e+001	1
15	3.000000e+000	5.600000e+001	1
16	2.000000e+000	5.600000e+001	1
16	1.000000e+000	5.600000e+001	2
17	2.000000e+000	5.600000e+001	1

Fig 5 Snapshot of Reliability of Schneidewind model

Analysis: Reliability graph for Schneidewind model is somewhat similar to the reliability graph of S-shaped model. Reliability in s-shaped model increased at high rate after some test intervals whereas in case of schneidewind model, variation in reliability is slow as compared to s-shaped model. Filters such as Hann and Round are applied. Hann filter removes the noise that appears in the set of failure data and Round filter is necessary in case of some models who accept the round off data as input.

C. For Littlewood Verrall model- RELIABILITY GRAPH BASED ON TIME BETWEEN FAILURE UNCERTAINTY FACTOR: CONSIDERED FOR LINEAR AND QUADRATIC LITTLEWOOD VERRALL.

Analysis: CASRE helps in generating the reliability graph when the time between failure information is available as input to the tool. X-axis shows the cumulative time between failures in minutes and Y-axis shows the reliability for some duration of time. This model accepts the three column format failure information as input and generates the reliability curve for the model. Some parameter estimation method is used by the model. Above generated graph show the reliability curve for both the linear and quadratic Littlewood verrall model. Reliability of quadratic LV increases at high rate as compared to the linear LV curve. As the time between failures increases, reliability increases at high rate.

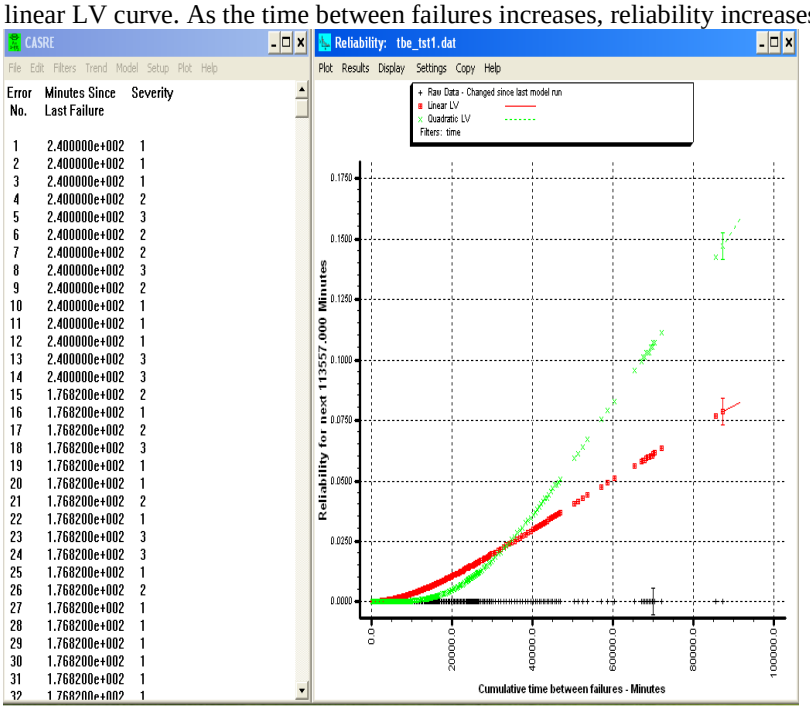


Fig 6 Snapshot of Reliability of Linear and Quadratic Littlewood Verrall

D. For Musa Basic Execution time model- Variation of Reliability graph for time between failure uncertainty factors.

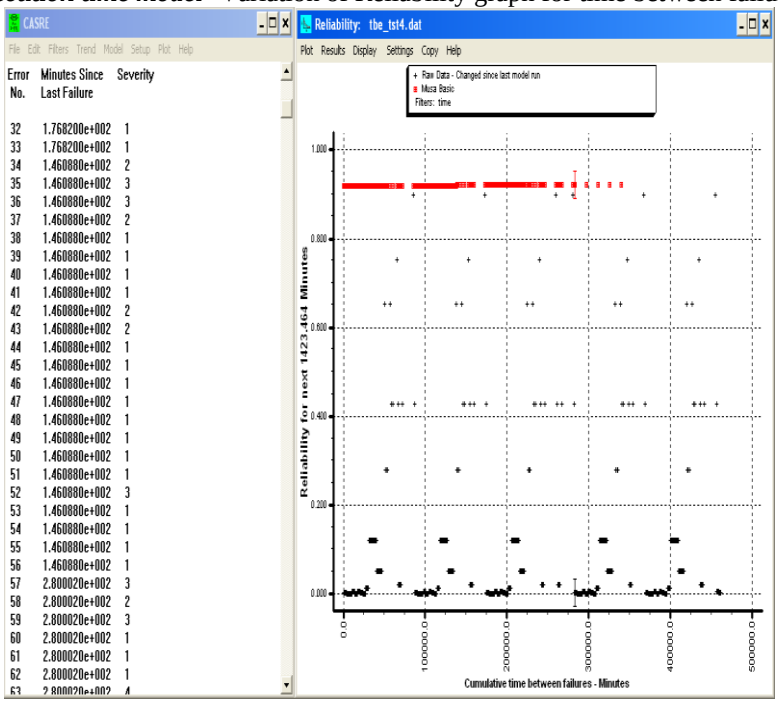


Fig 7 Snapshot of Reliability of Musa Basic Execution Time Model

Analysis: In case of Musa basic execution time model, X-axis show the cumulative time between failures in minutes and Y-axis show the reliability for the next duration of time. Here, reliability remains constant as the cumulative time between failures increases. Using the tool, time between failure data can be converted from one time unit to another. The above graph donot show any variation in the curve even after considering the time between failure information in hours. The reliability curve remains constant showing a stable growth of the model.

VI. ANALYSIS

A. Considering Number of Failures as Uncertainty Factor

From the graph for Schneidewind model we concluded:

- During the first few test intervals the reliability of Schneidewind model is constant. As the equal length test interval number increases the reliability estimation of model with respect to failure information increases.
- Failure decreases with same test interval executed more than once therefore increase in reliability curve.

Graph for S-Shaped model conclude that:

- Slope shows nearly same reliability prediction as above model w.r.t test interval number considering number of failures as uncertainty factor.
- S-shaped model provides more high reliability prediction than Schneidewind model for different failure data.

B. Considering Time between Failures as uncertainty factor

- Littlewood verrall model provide high reliability prediction for time between failure data as Compared to Musa basic execution time model.
- **Assumption**: Successive time between failures is independent.
- Reliability of Musa basic model will remain constant w.r.t cumulative time between failures with failure occurences having nearly equal time gap.
- **Assumption:** Number of failures that can be experienced in infinite time is finite**.** So the reliability graph shows the constant line for given TBF data.

VII. CONCLUSION

From comparative analysis of above models we concluded that:

- For S-shaped and schneidewind model – Reliability is inversely proportional to the Number of Failure uncertainty factor.

$$\textbf{Reliability } \alpha \quad \frac{1}{\textbf{Number of failure}}$$

- For Littlewood and Musa basic model [11] - Reliability is directly proportional to the cumulative time between failure uncertainty factors.

$$\textbf{Reliability } \alpha \quad \textbf{Cumulative time between failures}$$

VIII. FUTURE WORK

- More uncertainty factors can be considered for reliability estimation.
- Research can be extended for white box models to predict the reliability considering several factors related to the internal structure of system.
- No model is perfect for all type of failure data. Enhancements are required in the models to make them useful for all type of data.

REFERENCES

[1] Michael Grottke, "Software Reliability Model Study",IST-1999-55017,2001
[2] DACS Software Reliability datasets, http://www.thedacs.com.
[3] iM. Chen, M.R. Lyu, and E. Wong, "Effect of Code Coverage on Software Reliability Measurement", IEEE Transactions on Reliability, vol. 50, no. 2, June 2001, pp.165-170.
[4] Littlewood, B. & Verrall, J. "A Bayesian reliability growth model for computer software", Journal of the Royal Statistical Society, series C, Vol. 22, No. 3, 1973, 332.346.
[5] Musa, J.D. & Okumoto, K. "Software reliability models: concepts, classification, comparisons, and practice", Electronic Systems Effectiveness and Life Cycle Costing, Slowirzynski, J.K. (ed.), NATO ASI Series, F3, Springer Verlag, Heidelberg 1983,395.424
[6] Nikora, A., "Computer Aided Software Reliability Estimation User's Guide (CASRE)", Version 3.0, 2002.
[7] Yamada, S., Ohba, M., Osaki, S., "S-shaped reliability growth modeling for software error detection", IEEE Trans.Reliab. 12, 475–484, 1983.
[8] Schneidewind, N.F. "Software reliability model with optimal selection of failure data", software Engineering, IEEE Transactions, **Vol.** 19, Nov 1993.
[9] J.K. Ousterhout, Tcl and the Tk Toolkit, Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.
[10] Rabiner L.R., "A Tutorial on Hidden Markov Models", in Proceedings of the IEEE, vol. 77, pp. 257-286, 1989.
[11] Musa, J.D., Iannino, A. & Okumoto, K. "Software reliability-measurement, prediction, application. McGraw-Hill, 1987.
[12] Musa, J.D. "A Theory of Software Reliability and Its Application", IEEE Trans. Software Eng. 1 (1975), pp. 312 – 327.
[13] T. Keller, N.F. "Schneidwind, Successful application of software reliability engineering for the NASA Space Shuttle", Proc. IEEE International Symposium on Software Reliability Engineering (Case Studies), 1997,pp. 71–82.

[14] S. K. Chandran, A. Dimov, S. Punnekkat, "Modeling uncertainties in the estimation of software reliability – a pragmatic approach", 2010 Fourth IEEE International Conference on Secure Software Integration and Reliability Improvement, July. 2010.

[15] Xuemei Zhang, Hoang Pham, "An analysis of factors affecting software reliability", The Journal of Systems and Software 50 (2000) 43-56.

[16] Ritika Wason, P Ahmed and M.qasim Rafiq, "New Paradigm for Software Reliability Estimation", International Journal of Computer Applications 44(14):39-44, April 2012.

[17] Musa, J.D. "A Theory of Software Reliability and Its Application", IEEE Trans. Software Eng. 1 (1975), pp. 312 – 327.

