Volume 2, Issue 10, October 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# An Overview of Software Reliability Models

**Latha Shanmugam,**
*Associate Professor, Department of MCA*
*MVJ College of Engineering, Bangalore*
India

**Dr. Lilly Florence**
*Professor, Department of MCA*
*Adiyamaan College of Engineering, Hosur*
India

***Abstract- Software Reliability is an useful measure in planning and controlling the resources during the development process so that high quality software can be developed. Planning and controlling the testing resources via Software Reliability Measures can be done by balancing the additional cost of testing and the corresponding improvements in Software Reliability. It is also a useful measure for giving the user confidence about software correctness. In this paper, we are giving an overview of software reliability models.***

***Keywords-Software Reliability (SWR), Software Reliability Model, Classification.***

## I. INTRODUCTION

For critical business applications, continuous availability is a requirement, and software reliability is an important component of continuous application availability. Software Reliability can be viewed in many ways as follows.

**Binary Concept:** Reliability implies probability. A Program may contain no errors and its reliability is unity. If the program contains errors, then its reliability is zero.

**Collection of Design and Test Techniques**:

All the design methods, used for the design of improved software and all the testing techniques used for detection and correction of software errors so that the software in question is relatively error free.

**Probabilistic Measure**: It is assumed that for a given software, the duration of operation, design limits and user environment have been specified, Then the probability of successful operation. Non-occurrence of software errors under the above specified conditions gives the quantitative value of SWR and its value ranges from 0 to 1.

**Probability Weighed against the cost of failure to the user**: software is meant to perform failure free operations under specified environment and duration of times. If it fails, then the user of the software encounters loses of varying severity, depending upon the nature of failure. The probability of failure free operation of the software under the above specified conditions, weighed against the cost to the user of failure occurrence gives the measure of Software Reliability [3].

**SET Concept**: Let F be a class of faults, defined arbitrarily and T be a measure of relevant time, the unit of which is detected by application at hand. Then, the reliability of software package, with respect to the class of fault F and with respect to the metric T is probability that no fault of the class F occurs during the execution of the program for a prescribed period of relevant time.

**Expression of SWR**

The expression of SWR[4] is based on probability and is given by probability of success at time t, $P_s(t)$.

$R(t) = P_s(t) = 1 - P_f(t) = 1 - F(t)$

$= 1 - f(x)dx$

Where R(t) =Reliability of software at time t

$P_s(t)$ = Probability of failure

$P_f(t)$ = Probability of failure

F(t) = Failure distribution function

f(x) = Failure density function

Density function is not convenient for study of failure data and we normally consider hazard rate or failure rate Z (t). SWR in terms of hazard rate z(t) is given as

$R(t) = \exp[ - z(x)dx ]$

This expression shows that the reliability of software increased with passage of time [1].

**Constituents of SWR**

- Detection of software errors
- Correction of detected errors
- Prediction/Estimation / Measurement of SWR parameters

 

**SWR Parameters**

Parameters concerned with SWR are Reliability R (t), Mean Time to Failure MTTF, Number of errors remaining NR, Time required for debugging (g) [14].

## II. CLASSIFICATION OF MODELS

Software Reliability Models can be classified in two ways, one is based on Failure History and the other one is Data Requirements [5].

### A. Classification Based on Failure History

On the basis of failure history, the existing SWRMs can be grouped into four main classes as:

- Time Between Failure Models (TBF Models)
- Fault Count Models (FC Models)
- Fault seeding Models (FS Models)
- Input domain based models (IDB Models)

**A.1 TBF Models** In this class of models; the process under study is the time between failures. It is assumed that the time between $(i-1)^{th}$ and ith failures is a random variable, following a distribution whose parameters depend on the number of faults remaining in the program during this interval. Estimates of these parameters are obtained from the observed values of TBFs and then the parameters of SWR are obtained from the fitted models.

**A.2 FC Models** In FC Models, the random variable of interest is the number of faults (failures) occurring during specified time intervals. It is assumed that failure counts follow a known stochastic process. Usually a Poisson distribution with a time dependent will be discrete or continuous failure rate. The time can be calendar time or CPU time Parameters of the failure rate can be estimated from the observed values of failure counts and then the SWR parameters are obtained from the appropriate expression.

**A.3 FS Models:** A Program has unknown number of indigenous faults. To this, a known number of faults are seeded. The program is tested and observed number of seeded and indigenous fault is counted. By the method of MLE and Combinatory an estimate of the fault content of the program prior to seeding prior to seeding is obtained and then from this value SWR parameters are computed.

**A.4 IDB Models:** In this approach, a set of test cases is generated from the input covering the operational profile of the input. Usually the input domain is partitioned into a set of equivalent classes, each of which is usually associated with a program path. An estimate of the reliability of the program is obtained from the failures observed during execution of the above sample test cases.

**B. Classification Based On Data Requirements:** On the basis of data requirements the SWRMS can be grouped into two main groups as Empirical Models and Analytical Models.

**Empirical Models**: An Empirical SWR model develops relationship or a set of relationship between SWR measures and a suitable software metrics such as program complexity using empirical results available from past data. These relation(s), then can be used for measurement of SWR for which the requires identification of the appropriate SWR metrics and the development of the right type and form of relationships between the metrics and reliability measures. Models of this type are Miranda Model. Hallstead Model. Schneider Model.

**Analytical Models**: An analytical model requires some form of data gathered from software failures. It is based on fitting of a suitable distribution with required assumptions for simplicity on a set of data gathered during software testing and prediction of SWR parameters from the fitted distribution. Analytical modeling involves four steps.

i) Defining the assumptions associated with a software test procedure.
ii) Developing an analytical model based on the assumption and the test procedure.
iii) Obtaining parameters for the model using collected data.
iv) Using the model for performance prediction.

Analytical models can be further subdivided into Static Models and Dynamic Models based on time dependent behavior of collected data.

**B.1 Static Models:** This type of models, do not consider the time dependent behavior of software failures. They can be thought as discrete time models with one time interval. Depending on types of data used in the development of the models, the static models can be further subdivided into Error Domain Models (Combinational Model) and Data Domain Models.

**B.1.1 Error Domain Models:** Error domain models are static models developed using different set of errors, tagged or seeded. Models under this class are Mills Models, Lipow Model and Basin Model.

**B.1.2 Data Domain Models:** Data domain models are static models developed using different sets of input data and observed software failures. Model under this class Nelson Model.

 

**B.2 Dynamic Models:** Dynamic models represent the time dependent behavior of software failures. The Software failure data is collected over a period of time. Based on the time interval used, the Dynamic models are further grouped into continuous Time Models and Discrete Time Models.

**B.2.1 Discrete Time Models:** In discrete time models, the numbers of failures during the time intervals or during the stages of testing are recorded. This gives a discrete time representation software failures. The time interval may be fixed or random and accordingly Discrete Time Models are further subdivided into Random Time Interval Models and Fixed Time Interval.

**B.2.1.1 Random Time Interval D.T. Model:** In Random Time Interval Discrete Time Models, each interval is a stage in which sequences of tests are run and the numbers of failures are recorded. The data collected are number of test runs, number of failures and the length of each interval. Models of this type are Shooman Model and Lapadula Model.

**B.2.1.2 Fixed Time Interval D.T. Models:** In Fixed Time Interval D.T. Models the time intervals are of equal length. This type of models also assumes the number or errors detected during different time intervals are independent and have Poisson distribution. Models under this type are Moranda Geometric Poisson Model, Schneidewing NHPP Model.

**B.2.2 Continuous Time Models:** Data for these models are actual software failure times and hence give a continuous time representation of software failures. Based on the distribution function of the inter failure times the continuous time models can be further subdivided into independently Distributed Inter failure Times Models (IDT Models) and Independent and Identical Error Behavior Models (IIE Models).

**B.2.2.1 IDT Models:** In this type of models, the inter failures times (i=1,2,….m where m is total number of failures) is independently distributed and have similar distribution functions with different parameters.

**B.2.2.2 IIE Models:** In this type of Models the inter failure times are assumed to have identical and independent probabilistic behavior for each error Models under this class are Shantikumar Model and Binomial Model.

## III. BRIEF INTRODUCTION TO EXISTING SOFTWARE RELIABILITY MODELS

Software Reliability Models have been proposed on various assumptions and techniques. Some of the factors considered are program complexity, seeding techniques. Various distribution functions for inter failure time and number of failures etc. the models proposed range from simple empirical ones to the complex ones like Little wood & Verrall Model, where extensive use of Bayesian distribution and Gaming distributions have been made. The failure process itself is very complex involving interaction of human factors, program logic and input and output spaces which are very difficult to put in mathematical models. However attempts have been made in the design of SWRMS to approach to the real operational environment as possible. More than 200 Software Reliability Models were proposed by the software experts, among them few reliability models were discussed in the SWRMS

TABLE -1 **ABBREVIATION**

| S.No | Type | Model Description | Example Model |
|---|---|---|---|
| 1 | TBF | Time Between Models | J-M De-Eutrophication, Schnick and Wolverton, Goel and Okumoto Imperfect Debugging, Littlewood-Verall Bayesian Models |
| 2 | FC | Fault Count Models | Goel-Okumoto NHPP Model. Generalized Poisson Model, IBM Binomial and Poisson Models, Musa-Okumoto Logarithmic Poisson Execution Time Model. |
| 3 | FS | Fault Seeding Models | Mills seeding model, Lipow model, Basin model. |
| 4 | IDB | Input Domain based Models | Nelson Model, Ramamoorthy and Bastani Model. |
| 5 | IIE | Identical Error Behavior Models | Shantikumar Model and Binomial Model |
| 6 | IDT | Independently Distributed inter-failure models | JM Model, SchickWolertinModel,Moranda Model and Goel-Okumoto Model. |

 

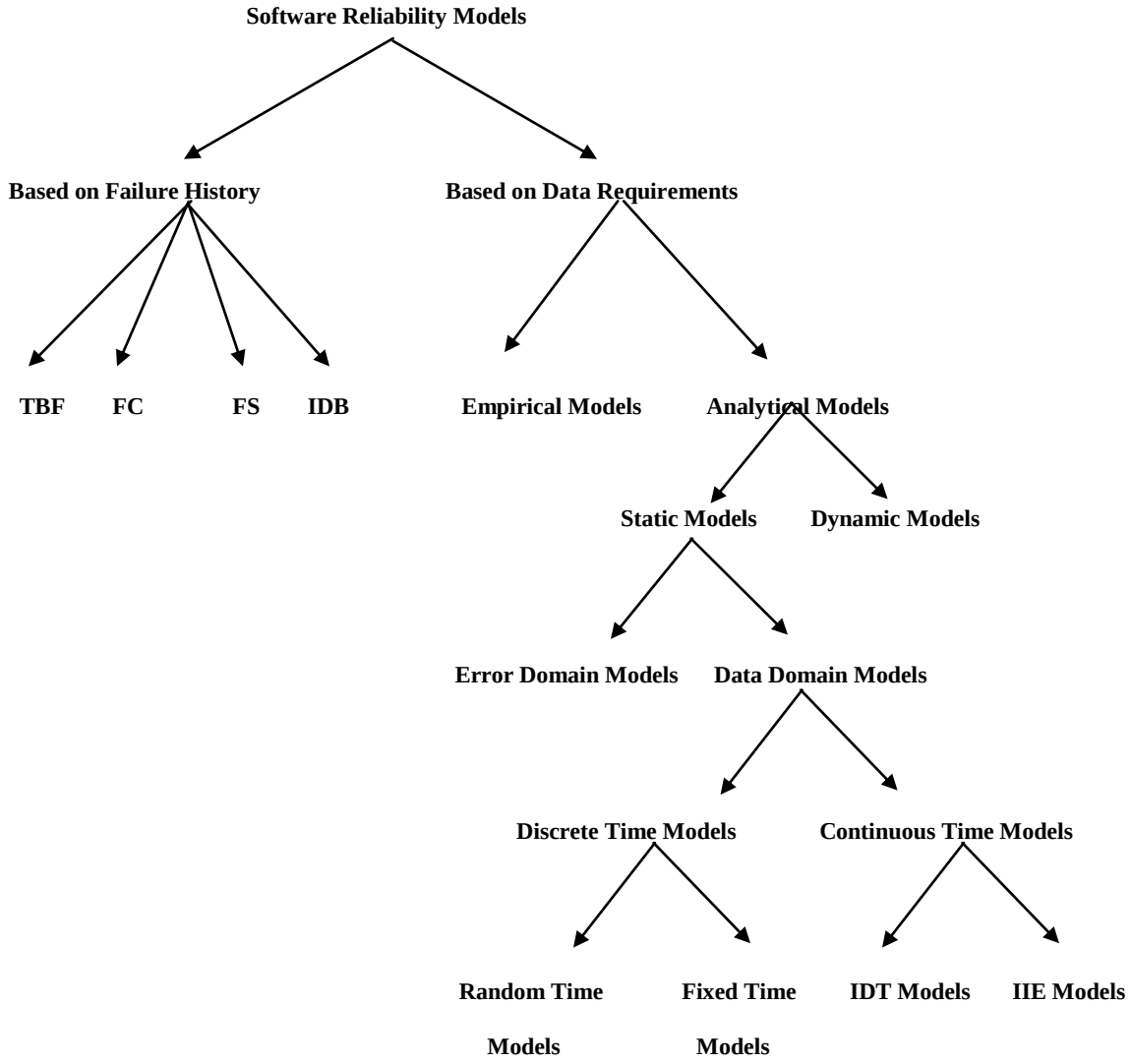


**Fig 1 Classification of software reliability model**

1. **Moranda Empirical Model**

   The software metric considered by these models is number of instructions. The empirical formula for the number of software failures, n, suggested by Moranda is $n = 1/50 \left( \sum_{i=1}^{m} wi/m \right)$,

   Where m = number of runs $w_i$ = number of instructions exercised in the $i^{th}$ run. However, no relationship for number of remaining errors, N(t) is given[5].

2. **Halstead Model**

   Software metrics considered are number of operations and operands in program, measure of effects required to create a program and number of mental discrimination between errors [6]. The empirical relation for number of software errors, N at the beginning of the test phase is given by: $N = KV / E_0 = E^{2/3} / E_0$,

   Where K = Constant

   V = number of bits required to specify a program & is given by $V = L\log_2 V$

 

L= the total number occurrences of operators and operands in a program,
v=the number of distinct operators and operands in the program
E0=mean number of mental discriminations between errors and its value is between 3000 to 32000
E = measure of efforts required to create a program and is given by E = V/L
l=program level = /v

3. **Schneider Model** The empirical estimation for software errors suggested for software development project is given by[7]
E(n) = 7.6 $E_{T0.006}$ * $S_{0.333}$ = s(S/s * 0.047) 1.667, where E(n) = expected number of software problem reports in the software development project. $E_T$ = Overall professional effort in "man months"
s = number of subprograms. S = Overall count of thousands of coded source statements of software.

4. **Mills Hyper Geometric Model**
This model make use of statistical procedure ie. , seeding technique / capture recapture sampling technique to assess the number of indigenous / unknown errors in a program. It is an errors domain model, assumes Hyper geometric distribution of errors, seeded or unknown[8]. This model avoids problem of time. A known number of errors, Ns are seeded in the program, increasing the number of errors from N0 to $N_0 + N_s$. The debugging process, is then continued and record of discovered unknown and seeded errors are made. The number of remaining unknown errors, n are then assumed to be a function of the ratio of discovered to remaining seeded errors and is given by $N = N_0 [ 1 – \exp( -n_s / ( N_s – n_s) ) ]$ Where, ns = number of seeded errors remaining. If debugging is done till all seeded errors are found, then the probability of finding m original errors is
$P(m) = (m + N_s – 1) C N_S – 1 / (N_s + N_0) C N_S$ m = 0, 1, 2, $N_0$ Then the maximum likelihood estimate subject to the constraint norm is N0 = m ie. , number of errors remaining in the software, $N_T = 0$

5. **Lipow Model** This is modified Mills model [9] by model by taking into consideration the probability, q of finding an error in each of the m tests. If this probability q is same for both actual and seeded errors, the probability of detecting n actual and n seeded errors is given by $P(n, ns) = m C_{an} + n_s . q^{n + ns} . (1-q)^{m-n-ns} * NC_n . N_s. C_{ns} / ( N + N_S) C_{n + ns}$
$N \geq n \geq 0$; $N \geq n_s \geq 0$ ; $M \geq n + n_s \geq 0$

6. **Nelson Model** This is a data domain model[10]. It assumes the following,
   1) Input profile distribution is known.
   2) Random Testing is used.
   3) Input domain can be partitioned into equivalent classes.

Reliability of software is measured by running the software for a sample of n set of inputs. These n inputs are chosen randomly from N mutually exclusive input domain subset, Ei, I = 1,2,3,…N ie each Ei is the set of data values needed to make a run. The random sampling of n inputs is done accordingly to a probability distribution. Pi. I = 1,2,…..N. This set is operational profile or user input distribution. If n is the number of inputs that resulted in execution failures than estimation of SWR is given by R.
$R1 = 1 - \frac{ne}{n}$

7. **Ramamoorthy and Bastani Model** This is an Input domain based model[11]. This pertains to critical, real time, process control program. In such systems no failures should be detected during the reliability estimation phase. So that the reliability estimated is one, hence the important metric of concern is the confidence in the reliability estimate.
This model provides an estimation of the conditional probability that the program is correct for all possible inputs given that it is correct for a specified set of inputs. The stochastic information about the behavior of the program for other points which are close to the test points. The main result of this model is
Prob (program is correct for all points in [ a , a + V] / It is correct for test cases having successive distances $x_j$ , j = 1,2,…2n-1
$e^{\lambda V} = \prod_{j=1}^{n-1} [ \frac{2}{1+e-\lambda x_j}$ , where x is a parameter deducted from some measure of the complexity of the source code and needs to be validated experimentally.

8. **Lapadula Model:** This is discrete time with random time interval model [12]. In this model, a sequence of test is conducted in M stages. Each stage terminates when any change is made to the software based on the number of errors detected during each stage, a reliability growth curve is fitted. The reliability of the software, R(i), during the ith stage is given by
R(i) = R( ) - A/i I = 1,2,3,…. Where A is growth parameter.

 

R( ) = lim R(i), is the limiting reliability of the software.

The model parameters A, R ( ) are solution to

$\frac{\sum_{i=1}^{m}[\,(Si-mi\,)}{Si} - R\,(\,) + \frac{A}{i}\,] = 0$

$$\sum_{i=1}^{m}\left[\frac{(Si-mi\,)}{Si} - R(\,) + \frac{A}{i}\left(\frac{1}{i}\right)\right\} = 0 \text{ where}$$

Si = number of tests during stage i
mi = number of failure during the $i^{th}$ stage
The reliability of the software in the $i^{th}$ stage is given by R (i) = R ( ) – A (i) } = 0 where
Si = number of tests during stage i
Mi = number of failures during the $i^{th}$ stage
The reliability of the software in the $i^{th}$ stage is given by
R(i) = R( ) – A(i) , I = m + 1, m + 2,…..

**9. Shantikumar's NHPP Model**

This is a continuous time-independent and identical error behavior model [13]. This is a simple Binomial model using non homogeneous Markov process representation of the number of software failures. The hazard rate function, (t) of an error is an arbitrary function which is to be provided by the analyst. Model parameters are parameters of (T) i.e, & b in this special case. Maximum likelihood estimates of these parameters are solution of the equations

1. $\sum_{i=1}^{m}\frac{1}{N-i+1} - \propto (\,1 - \exp(-\beta T) = 0$

2. $m/\propto \sum_{i=1}^{m}$ (N-I + 1) [ exp(-β t i – 1) - exp (-β t I ) ] =0

3. $m/\beta - \sum_{i=1}^{m} t\,i - \sum_{i=1}^{m} (N-i+1)\,[ti\,.\exp(\,-\beta\,t\,i-1) \propto = 0$

Where, $t_i$ = the time of the $i_{th}$ software failure
M = number of software failures.
Number of remaining errors, $N_T$ is given by $N_T$ = N-m
$R_T(t)$ is given by
$R_T(t)$ = (exp – N-m)) (exp(-βT) – exp(-βT) – βt), t ≥0

## IV. CONCLUSION

Software is an immovable mechanism that comprised of computer programs, procedures, rules, data and related documentation. The increase in number of software failures badly affected the performance of transportation, Telecommunication, military, industrial process, entertainment offices, aircrafts and business. Therefore software reliability has become more & more important. Reliability is the capability of software to maintain a determined level of performance within the time period. Software reliability is a measuring technique for defects that causes software failures in which software behavior is different from the specified behavior in a defined environment with fixed time . On the basis of the review the classification on software reliability models has been presented as a major contribution. This Classification is based on the various dimensions of reliability models. The major finding of the study is that the models under review reflect based on the failure data model and the data requirements model.

## ACKNOWLEDGEMENTS

The Authors would like to thank the Management, Principal Dr. K.S.Badarinarayan of MVJ College 0f Engineering, Bangalore for their moral support and guidance for doing research.

 

**REFERENCES**


1. The Hand Book of Software Reliability Engineering by Liu, Mc Graw Hill Publications.
2. Critical Review on Software Reliability by A. Yadav1 & R. A. Khan2 in International Journal of Recent Trends in Engineering, Vol 2, No. 3, November 2009. 114. *Critical Review on Software Reliability Models*
3. Software Reliability Engineering: Road Map By M i c h a e l R . Ly u
4. A critical review on software reliability modeling by Kai-Yuan Cai, Chuan-Yuan Wen, Ming-Lian Zhang Department of Automatic Control, Beijing University of Aeronautics & Astronautics, Beijing 100083, People's Republic of China
5. Jelinski. Z, Moranda P.B. "Software Reliability Research Statistical Computer Performance Evaluation", W. Frebierger, New York, Academic Press.
6. Use of the to bit model in contingent valuation: Experimental evidence from the Pemigewasset Wilderness Area by John M. Halstead, Bruce E. Lindsay, Cindy M. Brown Department of Resource Economics and Development, University of New Hampshire, Durham, New Hampshire 03824, U.S.A.
7. Schneiwind. N.F., "Methodology for software reliability prediction and quality control NTIS Report AD 754337. National Technical Information Service, Spring filed, V.A
8. Mills H.D., "On the development of large programs", Record of the 1973 IEEE Symposium on Computer Software Reliability, 1973.
9. Lipow., "A Branching Model with Population size dependence" Applied probability trust 1995.
10. Ground state of the massless Nelson model without infrared cutoff in a non-Fock representation, A Arai - Reviews in Mathematical Physics, 2001 - World Scientific.
11. Software Reliability—Status and Perspectives by CV Ramamoorthy, FB Bastani – IEEE Transactions on Software Engineering, 1982
12. A comment on the 'basic security theorem of Bell and LaPadula J McLean - Information Processing Letters, 1985 – Elsevier
13. A general software reliability model for performance prediction by JG Shanthikumar Microelectronics Reliability, 1981 – Elsevier
14. A Comparison of Parameter Best Estimation Method for Software Reliability Models by Latha Shanmugam, Dr. Lilly Florence in International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.5, September 2012.


 