International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

An Overview and Classification of Software Reliability Models

Dilip Sadhankar[1], Ashish Sasankar[2]
[1]Research Scholar, Department of Electronics & Computer Science, R.T.M. Nagpur University, Nagpur, Maharashtra, India
[2]Assocate Professor, Department of Computer Science, G.H. Raisoni Institute of Information Technology, Nagpur, Maharashtra, India

ABSTRACT

A software quality aspect is measured in terms of mean time to failure or failure intensity of the software. It is one of the key attributes when talk about software quality. Software quality may parts into quality aspect in various ways; however, software reliability seen as one of the key attribute of software quality. Software reliability is a valuable measure in planning and controlling the resources throughout the development process, as a result, high quality software can be developed. Scheduling and controlling the testing resources through software reliability measures can be completed by matching the additional cost of testing and the corresponding improvement in software reliability. It is too, a valuable measure for providing the user confidence about software correctness. A number of analytical models have been introduced in the past decades to assess the reliability of the software system. In this paper, researchers are giving an overview & analysis of software reliability models.

Keywords: Software reliability, Empirical model, Classification, times between failures, Estimation, failure count models, fault seeding, input domain models, model fitting, NHPP

I. INTRODUCTION

Reliability is usually defined in terms of a statistical measure; it is the probability to perform failure-free **software** operation for a specified period in a specified environment. The terms related to software reliability [1, 19].

- ✓ **Fault**: A condition causes the software to **fail** to perform its required function.
- ✓ **Error :** It refers to difference between Actual Output and Expected output
- ✓ **Failure** : It is the inability of a system or component to perform required function according to its specification

Software Reliability is difficult to achieve, since the complexity of software tends to be high. A failure corresponds to unexpected runtime behavior noticed/observed by a user of the software. A fault is a static software attribute, which results in a failure to occur. Faults need not necessarily results in failures. They only do so if the faulty component of the software is used [2].

A standard arrangement is proposed that can be applied to all phases of software development [4].

Permanent: failures occur for every single input values.

- **Transient:** failures occur only for definite inputs.
- **Cosmetic**: It may results into slight irritations. Do not lead to incorrect results.

- ✓ **Recoverable:** When failures take place, the system recovers with or without operator intervention.
- ✓ **Unrecoverable:** The system may require to be restarted.

II. MEASURING SOFTWARE RELIABILITY

Measuring software reliability remains a difficult problem, as people do not have a good understanding of the nature of software. Level of reliability needed for a software should be specified in the SRS document.

Numbers of models have emerged as people make an effort to understand the characteristics of how and why software fails, and try to quantify software reliability. Over 200 models have been developed since the early 1970s; however, how to quantify software reliability remains largely unanswered. Not a single models completely represent software reliability [5, 20].

Software modeling techniques can be divided into two subcategories:

- ✓ Prediction modeling
- ✓ Estimation modeling.

Both kinds of modeling techniques are based on observing and collecting failure data and analyzing with statistical assumption as shown below in table 1.

Two types of uncertainty causes to be any reliability measurement inaccurate:

- ✓ **Type 1 uncertainty: L**ack of knowledge with reference to how the system will be used
- ✓ **Type 2 uncertainty:** Reflects lack of knowledge about the impact of fault removal.

The majority of software models include the following parts:

- ✓ Assumptions,
- ✓ Factors,
- ✓ A mathematical function
- ✓ It relates the reliability with the factors.
- ✓ Is typically higher order exponential or logarithmic

III. CLASSIFICATION OF METHODS

Software reliability models classification is helpful in comparison of various reliability models. Different sets of models make it simple to obtain new models, which are more realistic than the existing ones by identifying the unrealistic assumptions made for these existing models. It also help managers/management to select a group of software reliability models based on their requirement.

Software Reliability Models can be classified using two approaches, first based on failure history and the second based on data requirements [6].

A. Classification Based on Failure History

Based on failure history, the existing software reliability models grouped into four main classes as [5]:

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

Figure 1 shows the classification of software reliability models.

1) **TBF Models:** In this class of models, the process under study is the time between failures. It is assumed that the time amid $(i-1)^{th}$ and i^{th} failures is a random variable, following a distribution whose parameters depend on the number of faults residual in the program during this gap. Estimates of these parameters obtained from the experimental values of TBFs and then the parameters of software reliability obtained from the fitted models.
2) **FC Models:** In FC Models, the random variable of significance is the number of faults (failures) taking place during specified time intervals. Generally, a Poisson distribution with a time dependent will be discrete or continuous failure rate. The time (CPU time or calendar time) parameters of the failure rate can be projected

from the observed values of failure counts and then the software reliability parameters are obtained from the suitable expression.

3) **FS Models:** A Program has unknown number of native faults. Apart from this, a known number of faults are seeded. The program is tested and observed number of seeded and native fault. By the method of MLE and combinatory an estimate of the fault content of the program prior to seeding is obtained and then from this value software reliability parameters are computed.
4) **IDB Models:** In this modeling approach, number of test cases generated from the input covers the operational profile of the input. An estimation of the reliability of the program is obtained from the failures observed through execution of the above sample test cases.

B. Classification Based On Data Requirements:

Based on data requirements the software reliability models can be divided into two main groups i.e. Empirical Models and Analytical Models.

1) **Empirical Models**: An Empirical software reliability model build up relationship or a set of relationship among software reliability measures and a appropriate software metrics such as program complexity. These relation(s) used for measurement of software reliability for which there is need of discovery of the appropriate software reliability metrics and the development of the precise type and form of relationships amid the metrics and reliability measures. Models of this type are Miranda Model, Hallstead Model and. Schneider Model.
2) **Analytical Models**: An analytical model needs some form of data elicited from software failures. This is based on fitting of an appropriate distribution with required assumptions for ease on a set of data collected while software testing and prediction of software reliability parameters from the fitted distribution. Analytical models can be further subdivided into Static Models and Dynamic Models based on time dependent behavior of collected data.
3) **Static Models:** This type of models, do not consider the time dependent behavior of software failures. It 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.
4) **Error Domain Models:** These types of models are static models developed using different set of errors. Models in this class are Mills Models, Lipow Model and Basin Model.
5) **Data Domain Models:** These types of models are static models developed with different sets of input data and observed software failures. Eg. Nelson Model.
6) **Dynamic Models:** This type of models represents the time dependent behavior of software failures. The Software failure data is collected over a period. Based on the time interval used, the Dynamic models are further grouped into continuous Time Models and Discrete Time Models.
7) **Discrete Time Models:** The numbers of failures in the time intervals or during the stages of testing are recorded in discrete time models. This presents a discrete time representation software failures. The time interval may be fixed or random and consequently Discrete Time Models are further subdivided into Fixed Time Interval and Random Time Interval Models.
8) **Random Time Interval D.T. Model:** In Random Time Interval Discrete Time Models, each interval is a stage in which sequences of tests run and the numbers of failures recorded. The data collected are number of test runs, number of failures and the length of each interval. Eg. Lapadula Model, Shooman Model.
9) **Fixed Time Interval D.T. Models:** In Fixed Time Interval D.T. Models, the time intervals are of the same length. This type of models also assumes the number or errors found in different

time intervals are independent and have Poisson distribution. Eg. Moranda Geometric Poisson Model, Schneidewing NHPP Model.

10) **Continuous Time Models:** An actual software failure time is the data for this model and therefore provides a continuous time representation of software failures. 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).
11) **IDT Models:** In this type of models, the inter failures times (i=1, 2,….m, where m is total number of failures) is separately distributed and have similar distribution functions with different parameters.
12) **IIE Models:** In this type of Models the inter failure times are assumed to have the same and independent probabilistic behavior for each error Models in this class are Shantikumar Model and Binomial Model.

Kindly refer table 2 for abbreviation and other information related with that.

IV. BRIEF INTRODUCTION TO EXISTING SOFTWARE RELIABILITY MODELS

Software Reliability Models have been proposed on various assumptions and techniques.. The failure process is very complex involving interaction of human factors, program logic and input and output spaces, which are very hard to put in mathematical models. However, attempts have been made in the design of software reliability models to approach to the real operational environment as possible. More than 200 software reliability Models were proposed by the software experts and amid them some reliability models were discussed below [7-9].

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

$$n = 1/50\,(\sum_{i=1}^{m} \frac{w_i}{m})$$

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 [10].

2) **Halstead Model:** Software metrics considered are number of operations and operands in program, measure of effects needed to create a program and number of mental bias between errors [11]. The empirical relation for number of software errors, N at the beginning of the test phase is given by:

$$\boldsymbol{N = KV/E_0 = / E^{2/3}/E_0}$$

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,

E_0=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 recommended for software development project is given by [8]

$$E(n) = 7.6E_{T0.006} * S_{0.333}$$
$$= s(S/s * 0.047)1.667$$

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) **Lipow Model:** This is modified Mills model by model taking into consideration the probability q of finding an error in each of the m tests [8]. 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

$P(n, ns) = m\ C_{an} + n_s.\ q^{\,n+ns}.\ (1\text{-}q)^{m-n-ns} * NC_n.\ N_s.\ C_{ns} / (N + N_s)\ C_{n+ns}$

$N \geq n \geq 0;\ N \geq n_s \geq 0;\ M \geq n + n_s \geq 0$

5) **Nelson Model:** This is a data domain model [7]. It assumes ,

1. Input profile distribution is known.
2. Random Testing is used.
3. Input domain can be partitioned into equivalent classes.

Reliability of software calculated by running the software for a sample of n set of inputs. These n inputs are selected randomly from N mutually exclusive input domain subset, E_i, i = 1, 2, 3…N i.e. each E_i is the set of data values needed to make a run. The random sampling of n inputs done in view of that 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 then estimation of software reliability is given by R.

$$R1 = 1 - \frac{n_e}{n}$$

6) **Ramamurthy and Bastani Model**

This is an Input domain based model [12]. This is based on critical, real time, process control program. In such systems, no failures should be detected during the reliability estimation phase. So that the reliability

Table 1. classification of modelling

ISSUES	PREDICTION MODELS	ESTIMATION MODELS
Data Reference	Uses historical data	Uses data from the current software development effort
When used in Development Cycle	Usually made prior to development or test phases; can be used as early as concept phase	Usually made later in life cycle(after some data have been collected); not typically used in concept or development phases
Time Frame	Predict reliability at some future time	Estimate reliability at either present or some future time

Table 2. Abbreviations for models

Sr. 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, Schick Wolertin Model, Moranda Model and Goel-Okumoto Model.

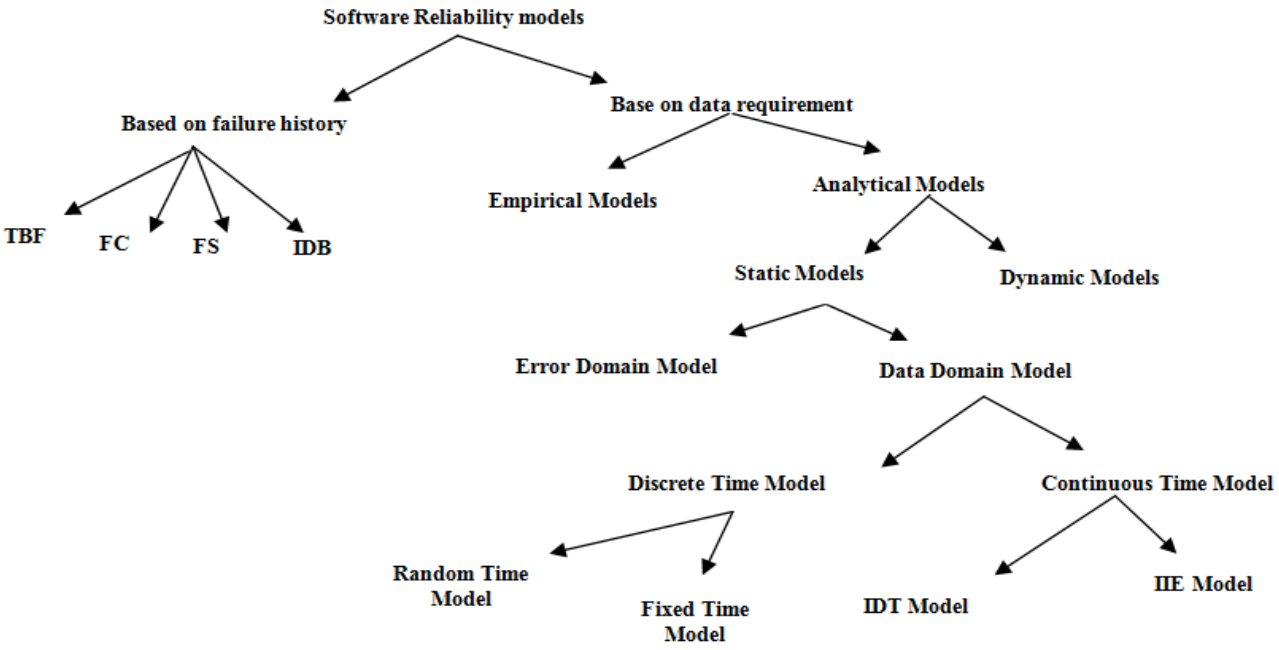

Figure 1. Classification of Reliability models

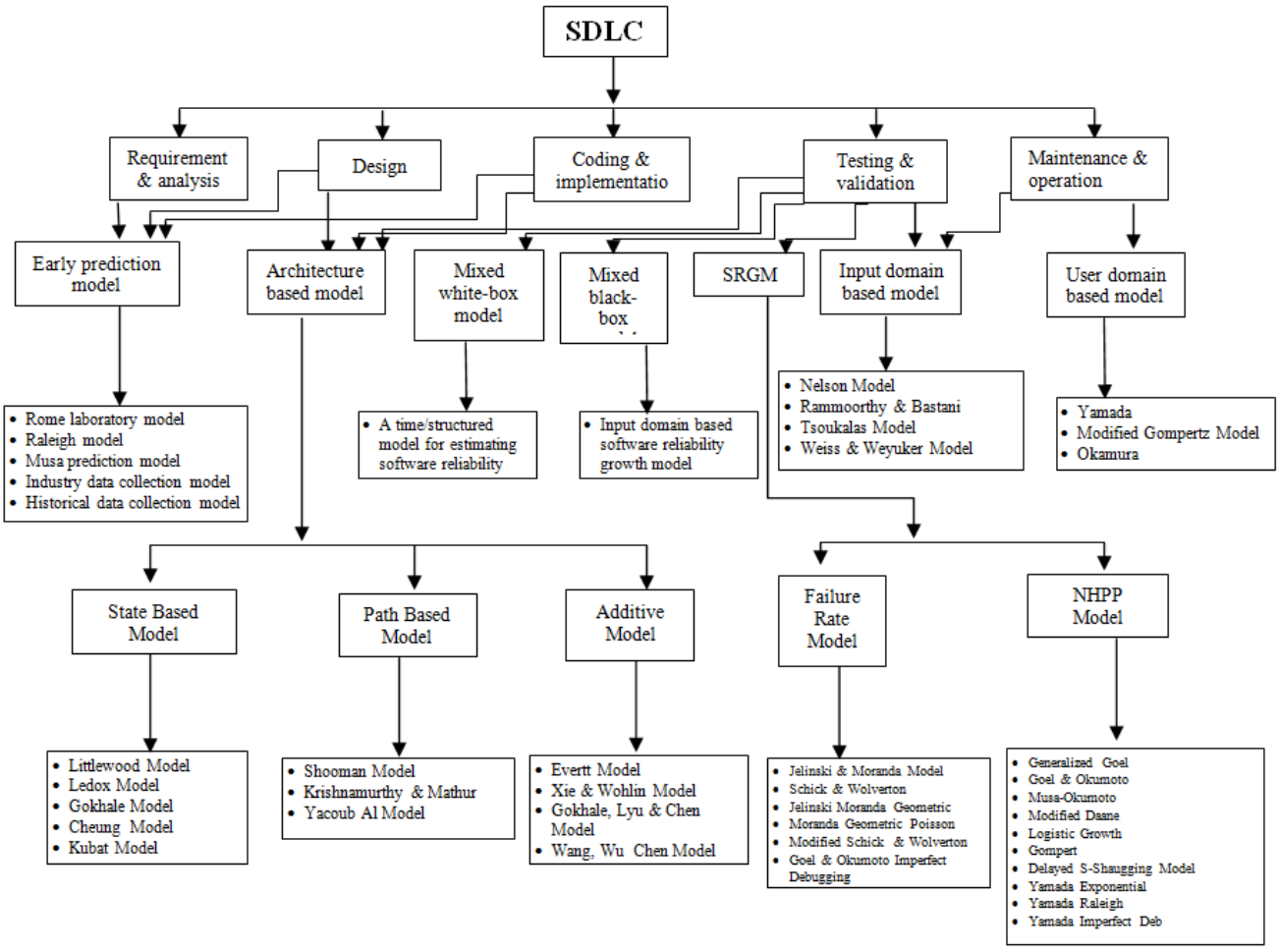

Figure 2. Software reliability models selection based on Software Development Life Cycle

V. CONCLUSION

Software is preset mechanisms that consist 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. Hence, software reliability has turn out to be more & more vital. Reliability is the ability of software to maintain a determine level of performance during time period. Software reliability is a measuring technique for defects which causes software failures in which software behavior is different from the specified behavior in a defined environment during fixed time. Based on the review, the arrangement on software reliability models has been presented as a major. This Classification is based on the various dimensions of reliability models. The key finding of the study is models under review based on the failure data model and the data requirements model.

In this paper, we also briefed about the applicability of these models on different phase of software development life cycle.

VI. REFERENCES

[1]. Lee, Kyoungwoo, Fault, Failure & Reliability

[2]. Michael R. Lyu, Software Reliability Engineering: A Roadmap

[3]. John B. Bowen, Standard error classification to support software reliability assessment

[4]. Zhi Wang, Bing Li and Yutao Ma, "An Analysis of Research in Software Engineering: Assessment and Trends"

[5]. Michael R. Lyu, Handbook of software reliability engineering

[6]. John D. Musa and Kazuhira Okumoto, "Software Reliability Models: Concepts, Classification, Comparisons, and Practice", Electronic Systems Effectiveness and Life Cycle Costing pp 395-423

[7]. A.L. Goel,"Software Reliability Models: Assumptions, Limitations, and Applicability",IEEE Transactions on Software, Volume SE 11,Issue 12,December 1985

[8]. Xie Min - 1991, Software reliability modeling

[9]. Dana Crowe, Design for Reliability

[10]. Z. Jelinski, P. Moranda, SOFTWARE RELIABILITY RESEARCH

[11]. N.E. Fenton ; M. Neil,"A critique of software defect prediction models",IEEE Transactions on Software, Volume: 25 Issue: 5

[12]. C.V. Ramamoorthy ; F.B. Bastani, "Software Reliability-Status and Perspectives", IEEE Transactions on Software, Volume: SE-8 Issue: 4

[13]. G.J. Schick ; R.W. Wolverton, "An Analysis of Competing Software Reliability Models", IEEE Transactions on Software, Volume: SE-4 Issue: 2

[14]. J.G.Shanthikumar, "A general software reliability model for performance prediction", Microelectronics Reliability, Volume 21, Issue 5, 1981, Pages 671-682

[15]. John D. Musa, A theory of software reliability and its application

[16]. H Pham, Software reliability

[17]. Latha Shanmugam and Dr. Lilly Florence, "An Overview of Software Reliability Models", Volume 2, Issue 10, October 2012 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering

[18]. A. Yadav1 & R. A. Khan, "Critical Review on Software Reliability", International Journal of Recent Trends in Engineering, Vol 2, No. 3, November 2009.

[19]. Jiantao Pan,"Software Reliability",Carnegie Mellon University 18-849b Dependable Embedded Systems Spring 1999

[20]. B. Parhami, Defect, Fault, Error,..., or Failure?, IEEE Transactions on Reliability (Volume: 46, Issue: 4, Dec 1997)

[21]. Jiantao Pan,"Software Reliability",Carnegie Mellon University 18-849b Dependable Embedded Systems Spring 1999