Indian Journal of Science and Technology, *Vol 8(29), DOI: 10.17485/ijst/2015/v8i29/85287, November 2015*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Classification of Software Reliability Models to Improve the Reliability of Software

G. Gayathry[1*] and R. Thirumalai Selvi[2]

[1]Department of Computer Science, Bharathiar University, Coimbatore – 641046, Tamil Nadu, India; gayathry.mgc@gmail.com

[2]Government Arts College (Men), Nandanam, Chennai - 600035, Tamil Nadu, India; sarasselvi@gmail.com

Abstract

Background/Objectives: The main objective is to present a classification of reliability models that would be useful in determining which of the existing model to use in a given software development environment. **Methods/Statistical Analysis:** In this research importance is given on comparison of existing software reliability models. The analytical models are mostly useful in estimating and monitoring reliability. The models can help software testing/debugging managers to make predictions about the anticipated future reliability of software under growth. **Findings:** This paper provides a detailed study of existing software reliability models which claim to progress software quality through efficient determination of software faults. Failure behavior of the software as predicted by these models has important implication in understanding the performance of the software and its improvement. One unique part of this paper is that we do not add any new models to the already large collection of models; rather we give importance to the taxonomy of models used in the software development process. **Application/Improvements:** The models are supportive for the software practitioners to master the schedule of the projects, the performance of the programmers and to improve the reliability of software system.

Keywords: Classification, Reliability Models, Software Reliability

1. Introduction

This paper highlights various Software Reliability Growth Models. SRGMs are statistical models which can be used to make predictions about a software system's failure rate, given the failure history behavior of the system. In practice such models are applied during the final testing phase when the development is virtually completed. Due to post-development testing, failures are identified and fixed in advance, the software becomes more stable and the reliability of software increases with time. SRGMs can be broadly categorized into two types (Pham, 2006). Deterministic one is used to study the number of distinct operators and operands and machine instructions in the program. Probabilistic one represents the failure occurrence and fault removal phenomenon of the testing process as probabilistic events with respect to time and testing effort[1].

Over the past 30 years, many SRGMs have been proposed for estimating reliability growth of products during the software development process[2]. Each model seems to work well with a particular data set, but no model appeared to do well for all data sets. Many researchers like Musa et al.[3] have shown that some families of models have, in general, certain characteristics that are considered better than others; for example, the geometric family of models tends to have better predictive quality than other models.

In[4] published a paper describing a non-homogeneous Poisson process model from the finite exponential class of models. This model was one of the first NHPP models proposed. This model predicted well on a unique data set.

In[5] proposed a NHPP based SRGM to describe various software failure/reliability curves. Both testing efforts and time dependent fault detection rate are considered for software reliability modeling. The applicability of proposed

*Author for correspondence

model is shown by validating it on software failure data sets obtained from different software development projects.

In[6] proposed a method for constructing SRGM based on NHPP. In this proposed method they have considered the case where the time dependent behaviors of testing-effort expenditure are described by Generalized Exponential Distribution (GED). SRGMs based on NHPP are developed which incorporates the (GED) testing-effort expenditure during the software testing phase.

In[7] first reviewed the logistic testing-effort function for modeling software reliability growth. They incorporate the logistic testing-effort function into S-shaped models. The proposed models are applied to two real data sets, to show that the logistic testing-effort function is more suitable for making estimations of resource consumption during the software testing phase.

2. Software Reliability Models

Software Reliability Models are used during software debugging process and it is mainly used to measure the quality of the software[8]. In this model, software is tested for a period of time, during which failures may occur. These failures cause a modification in design the new version of design is tested again. This cycle is repeated until design objectives are met. The paper discusses about the taxonomy of various software reliability models.

Figure 1 shows the hierarchy of software reliability models. At initial stage, the software reliability model is divided into two type's namely deterministic and probabilistic model. Two most common deterministic models are Halstead's software metric, based on unique number of operators and operands and McCabe's cyclomatic complexity metric based on cyclomatic number V (G). The probabilistic models include the following:

- Failure Rate Model (times between failure models).
- Failure or Fault Count Model (NHPP models).
- Error or Fault Seeding Model.
- Reliability Growth Model etc,.

2.1 Failure Rate Models

It is one of the earliest classes of proposed model used for estimating the reliability of software. In this model time between failures is taken for the process. The time between ith and (i-1)th failure data are taken for analysis. Fault data parameters are estimated from the observed

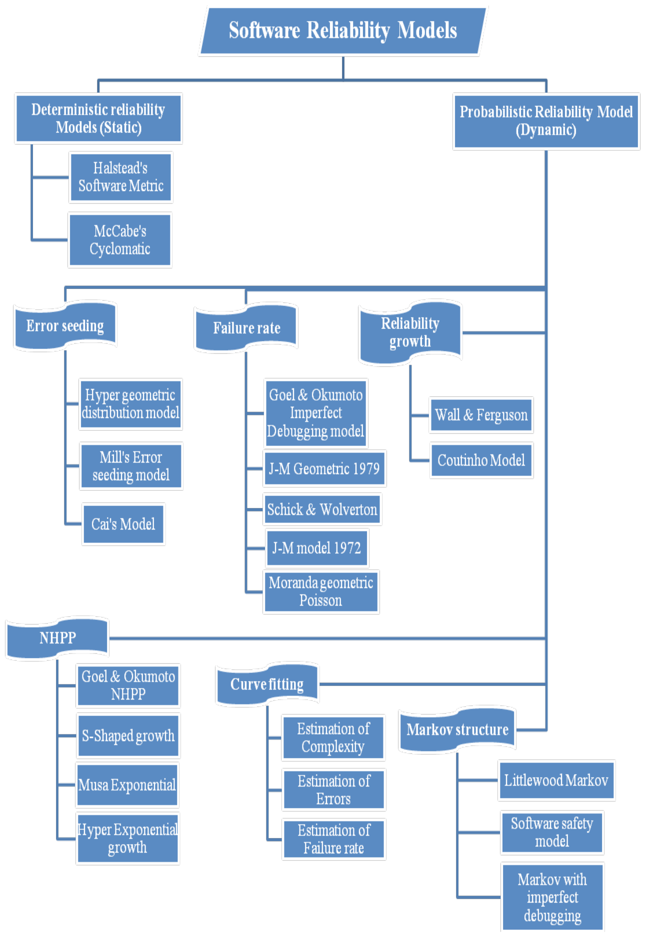

Figure 1. The hierarchy of software reliability models.

values of time between failures and estimates of software reliability, mean time to the next failure. The key models in this class are described in the following section.

2.1.1 Jelinski-Moranda Model

This is one of the most commonly used models for estimating software reliability[9]. During testing N independent software faults that may cause failures are taken for consideration. During the debugging process no new faults are introduced and the detected fault is removed in a negligible time. The hazard function also called as software failure rate is calculated during t1 time between the (i-1)th and ith failure is given by:

$$Z(t_i) = \varnothing\left[N-(i-1)\right]$$

Where Ø is proportionality constant. This hazard function is constant between failures, but when the fault is removed its size decreases.

2.1.2 Schick and Wolverton (SW) Model

The model is based on the assumptions of the JM model except that the hazard function is assumed to be proportional to the current fault content of the program as well as to the time elapsed is since the last failure[10] is given:

$$Z\,(t_i) = \varnothing\left[N-(i-1)\right]t_i$$

The hazard rate is linear with time within each failure interval. A modification of the above model is proposed in[11] whereby the hazard function is assumed to be parabolic in test time and is given by the function:

$$Z\,(t_i) = \varnothing\left[N-(i-1)\right]\,(-at_i^{\,2}+bt_i+c)$$

Where a, b and c are defined as constant. This function consists of two components, the first one is the hazard function of the JM model and the second one indicates that the reason for failure occurrence increases rapidly as the test time accumulates within a testing interval. At failure times ($t_i = 0$), the hazard function is proportional to that of the JM model.

2.1.3 Goel and Okumoto Imperfect Debugging Model

Goel and Okumoto[12] proposed an imperfect debugging model, which is the extension of the JM model. In this model, the number of faults in the system at time t, X(t), is treated as a Markov process whose transaction probability is governed by the probability of imperfect debugging. Time between the transitions of X(t) is taken to be exponentially distributed with rates dependent on the current fault content of the system. The hazard function during the interval between the (i-1)th and the ith failure is given by the function:

$$Z(t_i) = \varnothing\left[N-p(i-1)\right]\lambda$$

Where N is the initial fault content of the system, p is the probability of imperfect debugging and λ is the failure rate per fault.

2.1.4 Littlewood-Verrall Bayesian Model

In this model the times between failures are assumed to follow an exponential distribution but the parameters of this distribution are treated as a random variable with a gamma distribution that is:

$$F(t_i\,|\,\lambda_i) = \lambda_i e^{-\lambda i\ ti}$$

2.2 Fault Count Models

2.2.1 Musa Execution Time Model

In this model the reliability of software is analyzed based on execution time. The hazard function for this model is provided as:

$$Z\,(\tau) = \varnothing\, f\,(N-n_c)$$

Where τ is the execution time, f is the linear execution frequency, Ø is a proportionally constant fault exposure ratio and n_c is the corrected number of faults.

2.2.2 Goel-Okumoto NHPP Model

It is assumed that at random interval software system is subject to failures due to the faults in the present system. Let N(t) is the cumulative number of failures identified in time t then N(t) can be modeled as NHPP model for the following.

$$P\{N(t)=y\} = (m(t))^{y}\, e^{-m(t)},\ y=0,1,2.,$$
$$y!$$

$$m(t) = a\left(1-e^{-bt}\right)$$
$$\lambda(t) = m'(t) = abe^{-bt}$$

Where m(t) is the expected number of failures observed by time t, λ(t) is the failure rate, a is the expected number of failures observed and b is the fault detection rate per fault. In this case, the number of faults to be identified is treated as random variable whose values depend on the test factor.

2.2.3 Goel Generalized NHPP Model

During testing process, the failure rate is not constant it first increases and then decreases. Goel proposed a generalized NHPP model to handle the increasing/decreasing failure rate process.

$$P\{N(t)=y\} = (m(t))^{y}\, e^{-m(t)},\ y=0,1,2.,$$
$$y!$$
$$m(t) = a\left(1-e^{-bt}\right)$$

Where 'a' is the expected number of faults, b and c are constants that reflect the quality of testing. The failure rate is defined as follows:

$$\lambda(t) = m'(t) = abe^{-btc}t^{c-1}$$

2.2.4 IBM Binomial and Poisson Model

In these models, the fault detection rate is considered as a discrete process, and it follows a poisson or binomial distribution. It is assumed that the software system is developed and tested incrementally. These models can be applied to both module and system level testing.

2.2.5 Shoo Man Exponential Model

It is same as JM model. The hazard function for this model[13,14] is provided as:

$$Z(t) = k\left[(N/I) - n_c(\tau)\right)$$

Where t is the operating time of the system, I is the total number of instruction in the program, τ is the debugging time, $n_c(\tau)$ is the total number of corrected faults and k is the proportionality constant.

2.2.6 Generalized Poisson Model

The mean value function for this model is:

$$m(t_i) = \varnothing\left[N - M_{i-1})\right]\, t_i^{\alpha}$$

Where M_{i-1} is the total number of faults, Ø is a proportionality constant and α is a constant value used to rescale time t_i.

2.2.7 Musa-Okumoto Logarithmic Poisson Execution Time Model

In this model[15],the observed number of failures is assumed to be NHPP model and the mean value function is defined as :

$$\mu(\tau) = 1/\theta.\mathrm{Ln}(\lambda_0\theta_\tau + 1)$$

Where λ_0 represent initial failure intensity and θ represent the reduction in normalized failure intensity.

2.3 Fault Seeding Models

In this model, a known number of faults are 'seeded' in a program. The number of indigenous and seeded faults is counted at the time of testing. Using different estimation models, the number of indigenous faults and the reliability of the software are estimated.

2.3.1 Mills Seeding Model

It is one of the most popular and basic fault seeding models. To test a program, a random number of faults are seeded and then the program is tested for a particular time interval. The number of original indigenous faults can be estimated from the number of indigenous and seeded faults which are not taken into account at the time of testing. These models are also known as a tagging model since a given fault is tagged as seeded or indigenous.

2.3.2 Basin Model

Basin, suggested a two stage procedure with the use of two programmers which can be used to estimate the number of indigenous faults in the program.

2.3.3 Lipow Model

Lipow[16] proposed a model, which identifies the probability of fault in any test of the software. Then the probabilities of finding given number of indigeneous and seeded faults are calculated for independent tests.

2.4 Input Domain based Models

In this model, test cases are generated from an input domain. Partition of input domain into equivalence classes is a difficult task. The reliability is measured from the number of failures or execution of test cases.

2.4.1 Nelson Model

Nelson proposed a model[17], in which the reliability of software is measured by running the software for a sample of n inputs. The n inputs are randomly chosen from the input domain. The random sampling of n input is done through either probability distribution or simple through user input distribution.

2.4.2 Ramamurthy and Bastani Model

In this input domain based model[18], author mainly focuses on the reliability of real time, critical process control program. This model provides an estimate of conditional probability that the program is correct for all possible input given that it is correct for a specified set of inputs.

3. Conclusion

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. In this paper, various software reliability models are reviewed. Above analytical models are primarily useful in estimating and monitoring and it is viewed as a measure of estimation of software reliability and to enhance the quality of software.

4. References

1. Lyu MR. Handbook of software reliability engineering. NY: McGraw-Hill/IEEE Computer Society Press; 1996.
2. Zhang X, Pham H. An analysis of factors affecting software reliability. The Journal of Systems and Software. 2000; 50(1):43–56.
3. Musa JD, Iannino A, Okumoto K. Software reliability: measurement, prediction, and application. New York: McGraw-Hill Publication; 1987.
4. Goel AL, Okumoto K. A time-dependent error detection rate model for software reliability and other performance measure. IEEE Transaction on Reliability. 1979; R-28(3):206–11.
5. Kapur PK et al. Flexible software reliability growth model with testing effort dependent learning process. Applied Mathematical Modeling. 2008; 32(7):1298–307.
6. Quadri SM, Ahmad N, Farooq SU. Software reliability growth modeling with generalized exponential testing-effort and optimal software release policy. Global Journal of Computer Science and Technology. 2011; 11(2):27–42.
7. Huang CY, Kuo SY. Analysis of incorporating testing-effort function into software reliability modeling. IEEE Transaction on Reliability. 1979; 51(3):206–11.
8. Goel AL. Software reliability models: Assumptions, Limitations and Applicability. IEEE Transaction on Software Engineering. 1985; SE-11(12):1411–23.
9. Jelinski Z, Moranda P. Software Reliability Research, In Statistical Computer Performance evaluation. Freiberger W, Editor. New York: Academic; 1972. p. 465–84.
10. Schick GJ, Wolverton RW. An analysis of competing software reliability models. IEEE Transaction on Software Engineering. 1978; SE-4(2):104–20.
11. Schick GJ, Wolverton RW. An analysis of component software reliability models. IEEE Transaction on Software Engineering. 1982; SE-R:359–71.
12. Goel AL, Okumoto K. An Analysis of recurrent software failure in a real-time control system. Proc ACM Annual Tech Conf, ACM; Washington, DC. 1978. p. 496–500.
13. Chaurasia PK. Classification of Software Reliability Models. IJARCSSE. 2014 Aug; 4(8):1084–91.
14. Shooman ML. Probabilistic models for software reliability prediction. In: Freiberger W, editor. Statistical Computer Performance Evaluation. New York: Academic; 1972. p. 485–502.
15. Musa JD. A theory of software reliability and its application. IEEE Transaction on software engineering. 1971; SE-1:312–27.
16. Lipow M. Estimation of software package residual errors. TRW Redondo Beach, CA: Software Series. 1972; Rep-SS:359–71.
17. Nelson E. Estimating Software Reliability from test data. Micro electron. 1978; 17:67–74.
18. Ramamurthy CV, Bastani FB. Software reliability: Status and perspective. IEEE Transaction on Software Engineering. 1982 Jul; 8(4):359–71.