A Study on Comparison of Software Reliability Growth Models and Classification

B.ANNIPRINCY
Research Scholar
Sathyabama University,Chennai

Dr. S. SRIDHAR
Dean-Cognitive & Central Computing Facility
RV College of Engineering, Bangalore-560059

Abstract: Many architecture based software Reliability models has grown during the past decades in a very impressive manner. Several software reliability models have been discovered since the early 1970s and lots of work has been done on the models that estimate reliability growth during testing phase. Some of the major models are compared using an empirical case study. Software reliability model classification has been reviewed as a major contribution in this paper. This paper presents a review on software reliability models and the study on various dimensions and classification of reliability models.

KEYWORDS Software Reliability, Software Reliability Models, Models classification

I. INTRODUCTION

With increasing complication and emphasis on reuse, the current software engineering practice emphasizes development of component based systems. The existing black box models are clearly inappropriate to model such a large component based software system. Instead there is a need for a white box approach which estimates system reliability taking into account the information about the architecture of the software made out of components. The motivation for the use of architecture based software reliability approach includes the following:

- Understanding how the system reliability depends on its component reliabilities and their interaction
- Studying the sensitivity of the application reliability to reliabilities of components and interfaces
- Guiding the process of identifying critical components and interfaces for a given architecture
- Selecting an architecture that is most appropriate for the system under study.

Many architecture-based software reliability models have been proposed in the past, mostly by ad hoc methods. For extensive survey which proposes classification of the architecture based software reliability models, contains a detailed description of the key models, and discussion on the underlying assumptions, usefulness and limitations. Software Reliability growth models are designed to make predictions. Predictions of actual reliability or failure rate, time needed to reach a given reliability target and things like that. SRGM’s used for measuring the test quality, improving the test, finding critical points during testing etc. each SRGM has useful parameters (λ₀, the failure rate at the beginning of the test and N₀, the overall number of faults at the beginning of test). The models vary from “Generally optimistic to “Generally pessimistic” and most important, they do so in a continuous manner.

II. DIMENSIONS OF MODEL

MODEL DIMENSIONS

- Category : Number of Failures is infinite or finite
- Type : Distribution of the number of Failures experienced by the time specified.
- Class (only Finite Category): Functional form of the failure intensity over time.
- Time Domain: Calendar or Execution time.
- Family (only Infinite Category): Functional form of the failure intensity in terms of the expected number of failures experienced.
RELIABILITY MODELS CLASSIFICATION

<table>
<thead>
<tr>
<th>A. Bayesian Models</th>
<th>B. Infinite Failure Time Model</th>
<th>C. Weibull &amp; Gamma Failure Model</th>
<th>D. Exponential Failure Time Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Bayes Model</td>
<td>B. Infinite Failure Time Model</td>
<td>C. Weibull &amp; Gamma Failure Model</td>
<td>D. Exponential Failure Time Model</td>
</tr>
<tr>
<td>Reliability Growth Model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A. Bayesian Models

The models are used by several organizations like Motorola, Siemens & Philips for predicting reliability of software [3]. Bayesian incorporates past and current data. Prediction is done on the bases of number of faults that have been found and the amount of failure free operations.

L-V Reliability Growth Model: The model tries to account for fault generation in the fault correction process by allowing for the probability that the software program could become less reliable than before. For every correction of fault, a separate program has to write. Fault correction is obtained by fixing fault.

\[ D(i) = \mu(1/\beta(i)) \]

Where, \( \beta(i) \) = Sequence of independent variable.

D(i) = Distribution for the \( i^{th} \) failure.

B. Infinite Failure Time Model

Software is not completely error free when mean value function of a particular model tends to infinity. Models come under the category of Infinite Failure Time Model.

Duane’s Model: Duane observed that an erected line has been generated by comparing testing time with failure rate[1].

\[ \mu(t) / T = (\alpha T^4) T \]

Where, \( \mu(t) \) = Mean value function at time t.

\( \alpha T^4 \) = Expected Number of failures by time t.

\( T \) = Total testing

Geometric Model: In this model the time between failures is exponentially distributed and mean time failure decreased geometrically.

\[ E(Xi) = 1 / Z(ti-1) \]

Where, \( E(Xi) \) = Expected time between failure.

\( Z(ti-1) \) = fault detection rate.

Logarithmic Poisson Model: The model assumes that code has an infinite number of failures. The model follows NHPP. When failure occurs

distribution decreases exponentially. The possible number of failures over the time is a logarithmic function therefore it is called Logarithmic Poisson.

\[ \mu(t) = \alpha \exp(\mu(t)) \]

Where, \( \mu(t) \) = Mean value function at time t.

C. Weibull & Gamma Failure Models

Models under this category follow per fault failure Gamma distribution instead of exponential distribution.

Weibull Model: This model incorporates both increasing/decreasing and failure rate due to high flexibility[2]. This model is a finite failure model. MTTF = \( 1 - F(t) = \exp(\beta t^\alpha) \)

Where, MTTF = Mean Time to Failure

\( \alpha, \beta \) = Weibull distribution parameters.

\( t \) = Time of Failure

S-Shaped Model: This model considers that the number of failure within time period is a Poisson type model [4]. In this model time between failures depends on the time to failure. Mitigation of fault occurs immediately as failure happened.

\[ \mu(t) = q(1+ (1+ \beta t)e^{-\alpha}) \]

Where, \( \mu(t) \) = Mean value function at time t.

D. Exponential Failure Time Model

These models comprise of all finite failure models. Poisson and Binomial are two categorization of exponential Failure Time model [5]. The Binomial and Poisson types are based on per fault constant hazard rate. Hazards rate function is defined as the function of the remaining number of faults and the failure function is exponential.

\[ H(Z) = f(RNF) + f(exp(FF)) \]

Where, \( H(Z) \) = Hazard rate.

RNF = Renaming number of faults.

FF = Failure Function

J-M Model: The failure time is proportional to the remaining faults and taken as an exponential distribution. During testing phase the number of failures at first is finite. Concurrent mitigation of errors is the main strength of the model and error does not affect the remaining errors. Error removal is all human behaviour which is irregular also it cannot be avoided by introducing new errors during the process of error removal.

\[ MTBF = 1 / (N - (I - 1)) \]

Where, N = Total number of faults.

I = Number of fault occurrences.

MTBF = Mean Time between failure.

Time between the occurrence of the (i-1)
Execution Time Model: In this model the intensity function is directly proportional to the number of faults remaining in the program and fault correction is proportional to the number of fault occurrence rate.

Hyper Exponential Model: The idea behind this model is that the different parts of the software experience an exponential failure rate. However the rate varies through these parts to ponder different behaviours.

\[ \lambda(t) = N \sum \beta_i (\exp(-\beta_i t)) \]

Where, \[ \lambda(t) \] = Failure Intensity Function.
\[ T \] = Number of Failures.
\[ N \] = Finite Number of Failures.
\[ \beta_i \] = Total number of faults.

III. ANOTHER CATEGORIZATION

Classification based on Failure data

1. Time Between failure Models:
   - Jelinski- Moranda (Exponential Failure Model)
   - Musa- Basic Model (Exponential Failure Model)
   - NHPP (Exponential Failure Model)
   - Geometric (Infinite Failure Model)
   - Musa-Okumoto (Infinite Failure Model)
   - Littlewood – Verrall (Bayesian)

2. Failure Count Models:
   - Generalized Poisson.
   - Shick-Wolverton
   - Yamada S – Shaped (Gamma Failure Class)
   - NHPP (Exponential Failure Model)
   - Schneidewind (Exponential Failure Model)

Jelinski- Moranda Model:

This model follows exponential distribution but differs from the geometric model in that the parameter used is proposed to the remaining number of faults rather than a constant. All faults equally contribute to the reliability of the system.

Assumptions:
- The rate of fault detection is proportional to the current fault content of the software.
- The fault detection rate remains constant over the intervals between faults.
- A fault is corrected instantaneously without introducing new faults in the software.
- Every fault has the same chance of being encountered within a severity class as any other fault in that class.

Data requirements:
- Either actual failure time: \( t_1, t_2, \ldots, t_n \)
- Or elapsed time between failure: \( x_i = t_i - t_{i-1} \)

Musa-Basic Model:

Assumptions:
- The detections of failures are independent of one another.
- The software failures are observed (i.e., the total number of failures has an upper bound).
- The execution times (measure in CPU time) between failures are piecewise exponentially distributed.
- The hazard rate is proportional to the number of faults remaining in the program.
- The fault correction rate is proportional to the failure occurrence rate.

Non Homogeneous Poisson Process Model:

Assumptions:
- Assumes that the cumulative number of failures detected at any time follows a Poisson distribution.
- Time periods (intervals) can be unequal.
- Perfect debugging is assumed.

Data requirements:
- Fault counts on each testing interval \( f_1, f_2, \ldots, f_n \)
- Completion time of each period \( t_1, t_2, \ldots, t_n \)

Geometric Model:

The time between failures has an exponential distribution whose mean decreases in geometric fashion (i.e., the earlier faults have larger impact).

Assumptions:
- There is no upper bound on the total number of failures.
- All faults do not have the same chance of detection.
- The detections of faults are independent of one another.
- The failure detection rate forms a geometric progression and is constant between failure occurrence.

Data requirements:
- Either actual failure time: \( t_1, t_2, \ldots, t_n \)
- Or elapsed time between failure \( x_i = t_i - t_{i-1} \)

Musa-Okumoto Model:

Assumptions:
- Software is operated in a similar manner as the anticipated operational usage.
- Detections of failures are independent of one another.
- Expected number of failures is a logarithmic function of time.
- Failure intensity decreases exponentially with the expected number of failures experienced.
- There is no upper bound on the number of total failures (i.e., the program will never be error-free).
Data requirements:
- Either actual failure time: \( t_1, t_2, \ldots, t_n \)
- Or Elapsed time between failure: \( X_i = t_i - t_{i-1} \)

Weibull Model:
Assumptions:
- Total number of faults is bounded.
- The time to failure is distributed as Weibull distribution.
- The number of faults detected on each interval is independent for any finite collection of times.

Data requirements:
- Fault counts on each testing interval: \( f_1, f_2, \ldots, f_n \).
- Completion time of each period: \( t_1, t_2, \ldots, t_n \)

Yamada S- Shaped Model:
Assumptions:
- Cumulative number of faults follows a Poison Process.
- Total number of faults is bounded.
- Time between failures of the \((i-1)\) and the \(i^{th}\) depends on the time to failure of the \((i-1)\).
- When faults are detected they are removed immediately and no other fault is introduced.

Data requirements:
- Failure times \( t_1, t_2, \ldots, t_n \)
- Fault counts on each testing interval: \( f_1, f_2, \ldots, f_n \) together with the length of the interval.

Little Wood- Verrall Model:
Assumptions:
- There is no upper bound on the total number of failures, i.e., the program will never be fault free.
- The debugging may not be perfect, i.e., new bugs may be introduced during debugging.

Data requirements:
- Successive execution times between failures of independent random variables with exponential distribution with parameters \( 1, 2, \ldots, n \)

IV. RECENT DEVELOPMENTS

Several works have been done in this area. Some of them are as:
- In 2010, Kapil Sharma, Rakesh Garg, C.K. Nagpal, R.K. Garg proposed a framework for the Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach. They mentioned that Tools and Techniques for software reliability model selection found in the literature cannot be used with high confidence as they use a limited number of model selection criteria [8].
- In 2009, Du XianFeng, Qiang zanxia described Software reliability growth models based on non homogeneous poisson process
- In 2008, Leslie Cheung, Roshanak Roshandel, Nenad Medvidovic, proposed a framework for predicting reliability of software components at architectural design. Author identified reliability parameters and concludes the effects of reliability components. They mentioned the mechanism to overcome the lack of failure by using defect analysis and classification techniques, lack of operational profile information.[7].
- In 2007, Claes Wholin introduces three ways to estimate the parameters of the model. Author described that parameter can be evaluated by comparing historical data to previous data. Parameters can be estimated using information from the current project. He described that by unifying three approaches given by Claes Wholin with two parameters given by G.O Model, total number of failure is highly dependent on current project.

V. 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 and more important. Reliability is the capability of software to maintain a determined level of performance within the time period. Software reliability is the measuring technique for defect that causes software failures in which software behaviour is different from the specified behaviour in a defined environment with fixed time. On the basis of the review the taxonomy on software reliability models has been presented as a major contribution. This taxonomy is based on the various dimensions of reliability models. The major finding of the study is that the models under review reflect either infinite or finite number of failures. All exponential distribution based models reflect finite failures and logarithmic distribution based model reflect infinite failures.

REFERENCES


