www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 12 December, 2015 Page No. 15252-15257

Analysis of a Semi-Markov Model for Software Reliability

Ashish Namdeo[1] ,V.K. Pathak[2] & Anil Tiwari[3]

Comp-Tech Degree College, Sorid Nagar, Dhamtari(Chhattisgarh), India
B.C.S. Govt. P.G. College, Dhamtari PIN 493773, (Chhattisgarh), India
Disha College, Ramnagar Kota, Raipur, (Chhattisgarh), India

Abstract: In this paper analysis of a semi-markov model is done with reference to famous Jelinski-Moranda model which is probably the first model in software reliability. Fault removal resulting from the execution of program depends on the occurrence of the associated failure. Occurrence of failure depends both on the length of time for which the software has been executing and on the execution environment or operating condition. When different functions are executed, different faults are encountered and failures that are exhibited tend to be different.

Key Words: Semi-Markov Process, Decreasing Failure Intensity, Geometric De-Eutrophication

Markov Process Modelling :

Markov processes which are a general class of stochastic processes have been widely used and studied in Reliability Analyses. Many reliability models also belong to this category. A Markov Process is characterized by its state space together with the transition probabilities between these states.

A Stochastic Process $\{X(t), t \geq 0\}$ is said to be a Markov Process if its future development depends only on the present state of the process but not on the past i.e.

$$P[X(t) \geq x(t) \mid X(t_1) \geq x_1, \ldots\ldots\ldots\ldots, X(t_n) \geq x_n] = P[X(t) \geq x(t) \mid X(t_n) \geq x_n]$$

for all $t_1 < t_2 < \ldots\ldots t_n < t$.

The above property is generally called the Markov Property which has the following explanation: Given the present state of the process, its future behavior is independent of the past history of the process. This is the most important characteristic of a Markov Process. If the state space is discrete, a Markov Process is also called the Markov Chain. Let p_{ij} be the transition probability of the process between state i and j, i.e. $p_{ij}(t+s)=P[X(t+s) = j \mid X(s) = i]$, $s , t > 0$.

In general, p_{ij} may depend on t as well as on s. If all p_{ij}, $i, j > 0$ are independent of t, the Markov Chain is called time homogeneous.

The most famous result of homogeneous continuous time Markov Chain is that it satisfies the so called Kolmogorov equation, i.e. $p_{ij}(t+s)=\sum p_{ik}(s)\, p_{kj}(t)$, $s , t > 0$.

The theory of Markov Processes is well developed. The initial condition of the process

together with the transition probabilities completely determines the stochastic behavior of the Markov Process. Knowing the transition probabilities, the probability that the process is in a certain state can be can be obtained by solving Kolmogorov equations and other reliability measures can also be calculated. However, in order to get a mathematically tractable reliability model, some further assumptions usually have to be made.

The Jelinski-Moranda(JM) Model :

The best known software reliability model originally developed by Jelinski and Moranda in 1972 is also a Markov Process Model. It is one of the earliest models and has strongly influenced many later models which are in fact modifications of this simple model.

Model Assumptions and Some Properties:

The underlying assumptions of the JM model are :

1. The number of initial faults is an unknown but fixed constant.
2. A detected fault is removed immediately and no new fault in introduced.
3. The time between failures is independent and exponentially distributed random quantities.
4. All remaining faults contribute the same amount to the failure intensity.

Denote by N_0 the number of faults in the software before the test begins. By the assumption (3) and (4), the initial failure intensity is then equal to $N_0\varphi$, where φ is the constant of proportionality denoting the failure intensity contributed by each fault. It follows from the assumption (2) that, after a new fault is detected and removed, the number of the remaining faults is decreased by one. Hence after the k^{th} failure, there are (N_0-k) faults left and the failure intensity is decreased by $(N_0-k)\varphi$. Denote by T_i, i= 1,2,……N_0, the time between $(i-1)^{th}$ and the i^{th} failure, T_i is thus the i^{th} failure free time interval. By the assumptions, T_i's are then exponentially distributed random variables with parameter

$\lambda(i) = \varphi[N_0-(i-1)]$, i= 1,2,…….. , N_0 and the distribution of T_i is given by:

$P(T_i < t_i) = \varphi(N_0-i+1) \exp\{-\varphi(N_0-i+1)t_i\}$, i = 1, 2, ……$N_0$.

The main property of the JM-model is that the failure intensity is constant between the detection of two consecutive failures. This is quite reasonable if the software is unchanged and the testing is random and homogenous.

Decreasing Failure Intensity (DFI) models:

A serious critique of the JM- models that not all software faults are of same size. Some faults are more easily detected than the others. By incorporating this fact, some generalizations and modifications of JM-model are presented by Xie in 1987. The general formulation together with some special cases thereof is presented here.

A General DFI Formulation:

DOI: 10.18535/Ijecs/v4i12.29

The JM-model can be modified by using other jump function λ (i). Here, λ (i) is defined as the rate of occurrence of next failure after the removal of the $(i-1)^{th}$ fault. We define a failure intensity function λ (i) is DFI if λ (i) is a decreasing function of i. A DFI model is thus a Markov counting process model with decreasing failure intensity function. The general assumptions of the DFI Markov model are same as those for the JM-model. The failure counting process is assumed to be a Markov counting process. We also assume that all detected faults are immediately removed and no new faults are introduced during the testing. It should be noted here that these assumptions have lower order effects on software reliability and failure data can be usually modified in practice by counting the number of detected faults instead the number of failures, if a fault has caused more than one failures. Also, we assume that the test data are taken randomly from the input space of the software. Under the Markov assumptions, the times between failures are exponentially distributed with parameter λ (i).

Since software failure is a reliability growth process, by detecting and removing faults, the reliability does also increase. The failure intensity between the removals of two faults should also be constant, provided that the testing is random and homogeneous and the software is not subject to any change. However, the shape of the failure intensity as a function of removed faults may take different forms, depending on the inherent nature of the software. It can be observed that the failure intensity λ (i) for the JM-model is a linear function of the number of the remaining faults. In fact, since at the beginning, the big faults are likely to be detected, the decrease of the failure intensity is probably larger at the beginning than at the end of the testing phase. As a function of the number of the remaining faults, the failure intensity function is likely to be a convex function. Note that if all the software faults are removed, then the software will never fail. Hence, if a model for finite number of faults will be used, another general requirement on λ (i) is that $\lambda(N_0+1) = 0$ i.e. the failure intensity should be zero when the last fault has been removed.

Under the general assumptions above, the cumulative number of faults detected and removed, {N (t), t>0}, is a Markov Counting process with decreasing failure intensity λ (i). The theory of continuous time Markov chain can be applied. It can be shown that the collection of probabilities $\{p_i(t)=P[N(t) = i]; i = 0,1,2,\ldots\ldots, N_0, t\geq 0\}$ satisfies the Kolmogorov differential equations. The forward form of differential equations are given as follows:

$$p_0'(t) = -\lambda(1)\, p_0(t)$$

$$p_1'(t) = -\lambda(2)\, p_1(t) + \lambda(1)\, p_0(t)$$

.

.

$$p_i'(t) = -\lambda(i+1)\, p_i(t) + \lambda(i)\, p_{i-1}(t),\ i < N_0$$

and $p_{N_0}'(t) = -\lambda(N_0)\, p_{N_0-1}(t)$.

Furthermore, the assumption N(0) = 0 yields the following initial conditions:

$P_0(0) = 1$ and $p_i(0) = 0$ for $i > 0$.

The above Kolmogorov's differential equations can be easily solved and the solution is as follows:

$$p_0(t) = e^{-\lambda(1)t} \quad , \quad p_1(t) = \frac{\lambda(1)}{\lambda(2)-\lambda(1)}(e_1 - e_2) \quad ,$$

$$\ldots\ldots\ldots\ldots\ldots, \; p_i(t) = \sum_{j=0}^{i} A_j^{(i)} e_j \, , i \langle N_0$$

and for $i = N_0$, we have

$$p_{N_0}(t) = -\sum_{j=0}^{N_0-1} A_j^{(N_0-1)} \frac{\lambda(N_0)}{\lambda(j+1)} e_j \, ,$$

Where the quantities e_j, j = 0, 1,2,…….N_0-1 , are defined as $e_j = e^{-\lambda(j+1)t}$, j= 0, 1, 2,…..,N_0-1.

The quantities $A_j^{(i)}$ can be calculated recursively through

$$A_j^{(i)} = \frac{\lambda(i)}{\lambda(i+1)-\lambda(j+1)} A_j^{(i-1)} \, , j \langle i \quad \text{and}$$

$$A_i^{(i)} = -\sum_{j=0}^{i-1} A_j^i \, .$$

Generally, for a DFI model with parameter λ (i) , we have a set of parameters to be determined by using collected failure data.

Some Specific DFI Models:

Xie and Bergman in 1988 studied the power type DFI Markov model in which they assumed that the failure intensity λ (i) is a power-type function of the number of remaining faults, i.e.

$\lambda(i) = \varphi[N_0-(i-1)]^{\alpha}$, I = 1, 2 ,…… N_0.

Since λ (i) should decrease fast at the beginning and the decrease become slower for each i. Hence, it is reasonable to assume the λ (i) is a convex function of i and α is likely to be greater than one, since in this case, the decrease of the failure intensity is larger at the beginning. The exponential type Markov DFI model assumes that the failure intensity is an exponential function of the number of the remaining faults. It is characterized by the failure intensity function

$\lambda(i) = \varphi[e^{-\beta(N_0-i+1)}-1]$, i = 1, 2 ,…… N_0.

For the exponential type DFI model, the decrease of the failure intensity at the beginning is much faster than that at a later phase. The power type DFI model is a direct generalization of the JM-model which corresponds to the case α = 1. The parameter α can be treated as a new parameter and in this case, we have a three-parametric model. It is possible to reduce it to a two-parametric model by using a fixed value of α.

It is interesting to note that the exponential-type DFI model is similar to that of the Geometric De-Eutrophication model suggested by Moranda in 1975. Usually, software can never be completely fault-free and we may assume that there are an infinite number of faults in the software and the detection rate per fault φ is infinitely small. These assumptions together with the condition $\phi b^{N_0+1} \to a$, where a and b are unknown constants, gives $\lambda(i) = ab^{i-1}; i \geq 1$.

DOI: 10.18535/Ijecs/v4i12.29

This is just the failure intensity after removing i faults for the Moranda Geometric De-Eutrophication model. A similar model has been studied by Currit et al in 1986 where the mean time to next failure after m changes of the software is estimated by $MTTF_m = MTTF_0R^m$, where $MTTF_m$ is the mean time to failure after m changes. In the above model R and m are other model parameters.

Validation of the Exponentiality :

The power-type and exponential-type DFI model can be derived using aheuristic size-based sampling argument. Usually, a large fault corresponds to a larger failure probability and this means that larger faults are likely to be detected earlier than smaller faults are. The assumption that all faults have the same size is the most critical one for the JM-model.

Let M be the size of the input data, i.e. the number of data which can be used as input to the software. The quantity M is assumed to be large but finite for the sake of simplicity. One reason that M cannot be infinite in practice is that all data that can be stored in a computer are truncated and the number of data is limited. Note that an input datum can be multi-dimensional. Let M^* be the total number of these input data which may cause software failure. We also assume that M^* is much smaller that M since otherwise the software is too bad to be analyzed statistically.

Suppose that input data arrive at the software system according to a Poisson process with the intensity function ω which can be interpreted as the intensity of testing. Then the probability that the software encounters no failure in a time interval t is $1-F(t)=\sum_{j=0}^{\infty}(\frac{e^{-\omega t}(\omega t)^j}{j!})(\frac{M-M^*}{M})$ given M, M* and ω. According to Langberg and Singurwalla (1985) this has shock model interpretation. The first term inside the summation sign denotes the probability that j inputs or shocks are received during the time interval [0,t) while the second term inside the summation sign denotes that none of the j inputs causes software failure. If we let λ be defined as $\lambda=\frac{M^*}{M}\omega$ then it is easy to verify that $\sum_{j=0}^{\infty}(\frac{e^{-\omega t}(\omega t)^j}{j!})(\frac{M-M^*}{M})^j = e^{-\lambda t}$ and we have time to next failure has the following distribution $F(t)=1-e^{-\lambda t}$, which implies that if λ does not depend on t, then the time to next failure of the software is exponentially distributed with parameter λ. In this model the software testing remains valid, even in the case when all parameters depend on the number of removed software faults. Generally, we have that T_i, the time between the $(i-1)^{th}$ failure of the software, has a distribution function $F(t)=P[T\le t]=1-e^{-\lambda(i)t}$

It should be pointed out that usually λ(i) may also be a function of time. If this is the case, the exponentiality assumption is not valid and we have to use other distribution. However, if the system is specified and it does not suffer from any

great change, then the time dependence should be very weak, provided also that the test is homogeneous in the sense that the test intensity is constant on a reasonable scale such as man-power time or CPU-time whichever is suitable. Hence, we may assume that λ(i) is not time dependent and we have exponentially distributed time between failures.

References:

1. Jelinski, Z. and Moranda, P.B. [1972] "Software Reliability Research" Statistical Computer Performance Evaluation, Academia, New York, pp 465-484.
2. Shooman, M.L. and Trivedi, A.K. [1976], "A Many State Markov Model for Computer Software Performance Parameters", IEEE Transactions on Reliability, R-25(2), pp 66-68.
3. Currit, P.A.,Dyre, M. and Mills, H.D. [1986] "Certifying the Reliability of Software", IEEE Transactions on Software Engineering, SE-12(1), pp 3-11.
4. Langberg, N. and Singurwalla N. [1985] "A Unification of Some Software Reliability Models", SIAM Journal on Scientific and Statistics Computing, 6(3), pp781-790.