International Journal of Multidisciplinary Research and Development

IJMRD 2015; 2(2):37-42
www.allsubjectjournal.com
Received: 17-01-2015
Accepted: 05-02-2015
E-ISSN: 2349-4182
P-ISSN: 2349-5979
Impact factor: 3.762

Srisaila.A
Assistant Professor, Department of Information Technology, V.R. Siddhartha Engineering College Kanur, Vijayawada, Andhra Pradesh 520007, India.

R.SatyaPrasad
Associate Professor, Department of Computer Science & Engineering, Acharya Nagarjuna University Nagarjuna Nagar, Andhra Pradesh 522510, India.

M.V.D.N.S.Madhavi
Assistant Professor, Department of Mathematics, V.R.Siddhartha Engineering College Kanur, Vijayawada, Andhra Pradesh 520007, India.

Correspondence:
Srisaila.A
Assistant Professor, Department of Information Technology, V.R. Siddhartha Engineering College Kanur, Vijayawada, Andhra Pradesh 520007, India.

Assessing software reliability using HGM

Srisaila.A, R.SatyaPrasad, M.V.D.N.S.Madhavi

Abstract
As the usage of software is growing rapidly, accessing the software reliability is a critical task in development of a software system. So, many Software Reliability Growth Models (SRGM) are used in order to decide upon the reliable or unreliable of the developed software very quickly. In this paper, the well-known SRGM called as Hyper exponential Growth Model is adopted for time domain data based on Non Homogeneous Poisson Process (NHPP) which is used in accessing the reliability of developed software. The parameters are estimated using Maximum Likelihood Estimator.

Keywords: Software Reliability, Time domain data, Hyper exponential

1. Introduction

Today, almost everyone in the world is directly or indirectly affected by computer systems. Computers are used in diverse areas for various applications including air traffic control, nuclear reactors, aircraft, real-time sensor networks, industrial process control, automotive mechanical and safety control, and hospital health care, affecting many millions of people. An application of computer systems to the hospital health care is the monitoring of heart patients. In hospitals so equipped, sensors that detect electrical signals associated with heart activity are attached to the patient's heart area. The signals from these sensors are transmitted along wires to a computer programmed to analyze such data. If the incoming data indicate that the patient is doing well, the computer generated no output. If the data indicate the onset of serious conditions, the computer signals an alarm at the nursing station indicating which patient needs human care and the kind of help most apt to be useful.

Software can be defined as the collection of statements or language instructions are going to do some particular task. As the functionality of computer operations becomes more essential and yet more complicated and critical applications increase in size and complexity, there is a great need for looking at ways to quantify and predict the reliability of computer systems in various complex operating environments (Pham 2005c). Faults, especially with logic, in software design thus become more subtle. Usually logic errors in the software are not hard to fix but diagnosing logic bugs is the most challenging for many reasons [5]. The fault again is usually subtle.

An application of computer systems to the hospital health care is the monitoring of heart patients. In hospitals so equipped, sensors that detect electrical signals associated with heart activity are attached to the patient's heart area. The signals from these sensors are transmitted along wires to a computer programmed to analyze such data. If the incoming data indicate that the patient is doing well, the computer generated no output. If the data indicate the onset of serious conditions, the computer signals an alarm at the nursing station indicating which patient needs human care and the kind of help most apt to be useful.

As the functionality of computer operations becomes more essential and yet more complicated and critical applications increase in size and complexity, there is a great need for looking at ways to quantify and predict the reliability of computer systems in various complex operating environments (Pham 2005c). Faults, especially with logic, in software design thus become more subtle. Usually logic errors in the software are not hard to fix but diagnosing logic bugs is the most challenging for many reasons. The fault again is usually subtitle. In recent years software systems such as operating systems, control programs, and application programs have become more complex and larger than ever. It is quite natural to produce reliable software systems efficiently since the breakdown of the computer system, which is caused by software errors, results in tremendous loss and damage for social life. Then, software reliability is one of the key issues in modern software product development.

Many efforts have been developed to the study of measuring software reliability quantitatively in the area of software engineering [10].

2. Literature Survey

This section presents a detailed survey of the research literature that lead to carry out this research work grouped under the topics such as, Software reliability and their growth models, NHPP. This survey helped me to trace out some research problems for further investigation that formed my research work. Page (1954) developed CUSUM charts using Wald's sequential testing theory. Based on the notion of the distribution function of random variable Forman and Singpurwalla (1977) developed a probabilistic model describing the software failure phenomenon to suggest estimates of the parameters in the model and termination procedure for debugging the software.

Goel [12] and Okumoto [6] (1979) considered the probabilistic nature of software failure phenomenon based on an NHPP. They have analyzed the failure process to develop a suitable mean value function which in turn is used to get software performance measures. Musa [5] (1980) presented the need for potential use of software reliability measurement and made a comparison of software and hardware reliabilities. Ramamurthy and Bastani (1982) reviewed the status and perspectives of software reliability as on 1982. In Reckase (1983) and Spray (1993), they have done some numerical studies of the performance of the SPRT under tailored test setup for both the mastery and the multiple category criterion referenced tests.

Iannino *et al.* (1984) gave a descriptive narration of various criteria for the comparison of software reliability models based on predictive validity, quality of assumptions, capability, applicability, and simplicity. Matsumoto *et al.* (1988) discuss the evaluation procedure of a SRGM using data from a single program testing process applied to exponential, hyper exponential and Sshaped models, ranking the Sshaped model as superior with respect to estimation. Demmy and Petrini (1989) presented the major elements of SPC system, the steps required to apply SPC to software development activities and summarized the major advantages and disadvantages of SPC approach to software development. Fault density and failure intensity of an SRGM are used as two metrics to monitor software development capabilities and to measure customer satisfaction with the developed product in the research investigation of Huensch *et al.* (1990). Ehrlich *et al.* (1990) used the software reliability data collected during the testing of a system to measure the software quality in terms of experienced software failures. Malaiya *et al.* (1990) tried to smooth the noise by data grouping in preprocessing the failure data. They also tried some other methods like, windowing and data dependent grouping. Tohma *et al.* (1991) investigated six ways of the estimation of parameters in a hyper geometric distribution to get an estimate of number of initial faults in a program at the beginning of its testing debugging, along with their relative accuracies. Sofer and Miller (1991) used a nonparametric method of estimating the software failure rate in completely monotone models which can be compared with parametric approaches. Vallee and Ragot (1991) demonstrated the application of NHPP approach to the industrial world generating accurate predictions with specific applications in space research.

Stott (1991) discusses the use of Shewart control charts for monitoring the defect density at each stage of software development. Lantzy (1992) examined some of the principles of statistical process control that have been successfully applied to a variety of manufacturing processes and offers a set of transformations on these principles that permit their application to software process. Khoshgoftaer *et al.* (1992) suggested an SRGM with the help of curve fitting techniques, predicting the number of faults in a system through fitting nonlinear regression models to the number of faults in a program module. Zhao and Xie (1992) proposed a time dependent delay function, by arguing that detected faults become harder to correct with test in effect. Downs and Scott (1992) quantified and compared the performance of software reliability models with respect to a measure that helps to prefer one model to the other.

Liu (2011) proposed a function based nonlinear least squares estimation method which extends the potential fitting functions of traditional least square estimation in estimating parameters of Jelinski morando reliability model. Kulldorff *et al.* (2011) proposed a maximized SPRT based on a composite alternative hypothesis, which works well across a range of relative risks. They illustrated the use of this method on vaccine safety surveillance and compared it with the classical SPRT.

2.1 Software Reliability Growth models

Growth models are an important class of Software Reliability models. They were proposed in order to represent the process of finding and removing faults. These models were introduced in order to fit the failures production curve, on the other hand, past reports of failures production is used to predict the remaining number of failures or to find the failure rate. In a not intended to be exhaustive list, models usually mentioned in the literature of SRGM are: Goel Okumoto, S-shaped, Musa-Okumoto, Log-Logistic, Jelinski-Moranda. SRGM models can be applied during several phases of software development. In an advanced stage of software development, the predicted failure rate will be the starting rate at the testing phase. The prediction of the rate at the end of the testing phase will also be quite important since it is the beginning of the maturity phase, previous to deployment. The rate predicted at the end of the maturity phase will be that of the product at delivering.

In summary, Software Reliability Growth models (SRGM), can be applied in any stage of software development, all of them are very important. As the product becomes mature, the failure rate gets lower. This behavior is what the SRGM try to predict. What we want to obtain is either a prediction of the number of failures in a given future time interval, or the failure rate at a given time, for example at the moment of delivery, when the software will be in production. This information is useful to know whether the product is ready for release or not, how much testing will be needed to add, etc.

The inflection S shaped model (Ohba 1984) is based on the dependency of faults by postulating the following assumptions:

Some of the faults are not detectable before some other faults are removed.

1. The probability of the failure detection at any time is proportional to the current number of detectable faults in the software.
2. Failure rate of each detectable fault is constant and identical.
3. The isolated faults can be entirely removed.

2.2 Hyper Exponential Growth Model

The hyper exponential growth model (Ohba 1984a) is based on the assumption that a program has a number of clusters of modules, each having a different initial number of errors and a different failure rate. Examples are new modules *vs* reused modules, simple modules *vs* complex modules, and modules which interact with hardware *vs* modules which do not. It should be noted that the sum of exponential distributions becomes a hyper exponential distribution.

2.3 Maximum Likelihood Parameter Estimation

The method of maximum likelihood estimation (MLE) is one of the most useful techniques for deriving point estimators. Parameter estimation is of primary importance in software reliability prediction. Once the analytical solution for *m* (*t*) is known for a given model, the parameters in the solution need to be determined. Parameter estimation is achieved by applying a technique of MLE, the most important and widely used estimation technique. In many cases, the maximum likelihood estimators are consistent and asymptotically normally distributed as the sample size increases (Zhao 1996). In this chapter, we only discuss the MLE technique to estimate the unknown parameters for the software reliability models. Depending on the format in which test data are available, two different approaches are frequently used. A set of failure data is usually collected in one of two common ways. If we conduct an experiment and obtain N independent observations, t_1, t_2……., t_N, then the likelihood function is given by the following products:

$$L(t_1,t_2,t_3,.....,t_N/\theta_1,\theta_2,\theta_3,.....,\theta_k)=\prod_{i=1}^{N} f(t_i/\theta_1,\theta_2,\theta_3,.....,\theta_k)$$

Likely hood function by using λ (t) is:

$$L=\prod_{i=1}^{N}\lambda(t_i)$$

The logarithmic likelihood function is given by:

$$\wedge=\ln L=\prod_{i=1}^{N}\ln\ f(t_i/\theta_1,\theta_2,\theta_3,.....,\theta_k)$$

$$\log L=\log\left(\prod_{i=1}^{N}\lambda(t_i)\right)$$

which can be written as $\sum_{i=1}^{n}\log\left[\lambda(t_i)\right]-m(t_n)$

The maximum likelihood estimators (MLE) of $\theta_1,\theta_2,\theta_3,.....\theta_k$ are obtained by maximizing L or Λ, where Λ is in L. By maximizing Λ, which is much easier to work with than L, the maximum likelihood estimators (MLE) of $\theta_1,\theta_2,\theta_3,.....\theta_k$ are the simultaneous solutions of K equations such that: $\frac{\delta(\wedge)}{\delta\theta_j}=o,\ j=1,2,3,...,k$

The parameters 'a' and 'b' are estimated using iterative Newton Raphson Method, which is given as

$$x_{n+1}=x_n-\frac{g(x_n)}{g'(x_n)}$$

3. Proposed Work

3.1 Illustrating the MLE Method

3.1.1 Parameter estimation

To estimate 'a_i' and 'b_i', for a sample of n units, first obtain the likelihood Function:

$$L=\prod_{i=1}^{N}\left(\sum_{i=1}^{n}\left(a_ib_ie^{-b_it_i}\right)\right)$$

Taking the natural logarithm on both sides. The log likelihood function is given as:

$$\log l=\log\left[\prod_{i=1}^{N}\lambda(t_i)\right]=\log\left[\prod_{i=1}^{N}\left(\sum_{i=1}^{n}\left(a_ib_ie^{-b_it}\right)\right)\right]$$

Taking the partial derivative again with respect to 'a' and equating to '0'.

(i.e. $\frac{\delta\log L}{\delta a}=0$)

We will get the value of a_i as: $a_i=\frac{1}{\left(1-e^{-b_it_n}\right)}$

Taking the partial derivative again with respect to 'b' and equating to '0'.

(i.e. $g(b)=\frac{\delta\log l}{\delta b}=0$)

$$g(b_i)=\frac{\delta l}{\delta b_i}=\sum_{i=0}^{n}\left[\frac{1}{b_i}+\frac{\left(-t_ne^{-b_it_n}\right)}{1-e^{-b_it_n}}-t_i\right]=0$$

Taking the partial derivative again with respect to 'b' and equating to '0'.

(i.e. $g'(b)=\frac{\delta^2\log l}{\delta b^2}=0$)

$$g'(b_i)=\sum_{i=0}^{n}\left[\frac{\left(t_n{}^2e^{-b_it_n}\right)}{\left(1-e^{-b_it_n}\right)^2}-\frac{1}{b_i{}^2}\right]=0$$

The parameter 'b' is estimated by iterative Newton Raphson Method using

$b_{n+1}=b_n-\frac{g(b_n)}{g'(b_n)}$ .This is substituted in finding 'a_i'.

4. Results and Observations

The procedure of generating the reliability of given five software data sets as given below. The results have given the positive recommendations that software reliability can be assessed and the failures are detected at early stage.

Table 1: NTDS Software failure data reported by Jelinski and Moranda (1972)

Error Number N	Time between Errors Sk days	Cumulative Time Xn = ∑ Sk days
1	9	9
2	12	21
3	11	32
4	4	36
5	7	43
6	2	45
7	5	50
8	8	58
9	5	63
10	7	70
11	1	71
12	6	77
13	1	78
14	9	87
15	4	91
16	1	92
17	3	95
18	3	98
19	6	104
20	1	105
21	11	116
22	33	149
23	7	156
24	91	247
25	2	249
26	1	250

Table 2: AT & T Software failure data (1993)

Failure Index	Inter-failure time	Cumulative Inter - Failure time
1	5.5	5.5
2	1.83	7.33
3	2.75	10.08
4	70.89	80.97
5	3.94	84.91
6	14.98	99.89
7	3.47	103.36
8	9.96	113.32
9	11.39	124.71
10	19.88	144.59
11	7.81	152.4
12	14.6	167
13	11.41	178.41
14	18.94	197.35
15	65.3	262.65
16	0.04	262.69
17	125.67	388.36
18	82.69	471.05
19	0.46	471.51
20	31.61	503.12
21	129.31	632.43
22	47.6	680.03

Table 3: IBM Software failure data reported by Ohba (2005)

No of Errors	Inter failure time	Cumulative Inter failure time
1	10	10
2	9	19
3	13	32
4	11	43
5	15	58
6	12	70
7	18	88
8	15	103
9	22	125
10	25	150
11	19	169
12	30	199
13	32	231
14	25	256
15	40	296

Table 4: LYU Software failure data reported by Michale R (1996)

Failure No.	Time Between Failures	Cumulative Data
1	0.5	0.7
2	1.7	2.2
3	4.5	6.7
4	7.2	13.9
5	10	23.9
6	13	36.9
7	14.8	51.7
8	15.7	67.4
9	17.1	84.5
10	20.6	105.1
11	24	129.1
12	25.2	154.3
13	26.1	180.4
14	27.8	208.2
15	29.2	237.4
16	31.9	269.3
17	35.1	304.4
18	37.6	342
19	39.6	381.6
20	44.1	425.7
21	47.6	473.3
22	52.8	526.1
23	60	586.1
24	70.7	656.8

Table 5: XEI Software failure data (2002)

Failure No.	Time Between Failures	Cumulative Data
1	30.02	30.02
2	1.44	31.46
3	22.47	53.93
4	1.36	55.29
5	3.43	58.72
6	13.2	71.92
7	5.15	77.07
8	3.83	80.9
9	21	101.9
10	12.97	114.87
11	0.47	115.34
12	6.23	121.57
13	3.39	124.96
14	9.11	134.07
15	2.18	136.25
16	15.53	151.78

17	25.72	177.5
18	2.79	180.29
19	1.92	182.29
20	4.13	186.34
21	70.47	256.81
22	17.07	273.88
23	3.99	277.87
24	176.06	453.93
25	81.07	535
26	2.27	537.27
27	15.63	552.9
28	120.78	673.68
29	30.81	704.49
30	34.19	738.68

The above all are the datasets which are given as the inputs to estimate the parameters in the hyper exponential growth model. In hyper exponential growth model which we considered, have only two parameters a and b. These parameters are estimated by using Maximum likelihood estimator which is shown below:

Table 6: Parameter a & b values.

Datasets	Estimated Parameters	
	a	b
NTDS	2.1517	0.0033
AT & T	1.2235	0.0052
IBM	1.9142	0.0038
XIE	1.1876	0.0055
LYU	1.2406	0.0049

The reliability function R(s/x) is given by

$$R(s/x) = e^{-a[e^{-bx} - e^{-b(x+t)}]}$$

After substituting the estimated parameters a and b in the reliability function, then the calculated reliabilities are shown below

Table 7: Reliabilities using HGM.

Datasets	S	X	Reliability
NTDS	250	50	0.8666
AT & T	680.03	262.65	0.9754
IBM	296	88	0.8382
Xie	738.68	256.81	0.9846
Lyu	656.8	586.1	0.9542

5. Conclusion and Future Work

Software reliability is an important quality measure that quantifies the operational profile of computer systems. In this paper, we considered hyper exponential growth model which is a software reliability growth model. This model is primarily useful in estimating and monitoring software reliability, viewed as a software quality. Software Reliability is assessed with Hyper exponential growth model by estimating the parameters using Maximum likelihood method for time domain data. This analysis shows that AT&T data is more reliable than all other considered datasets. This work can be extended by applying Software Process Control (SPC) for time domain data and SPRT also.

6. References

1. Agresti A. Categorical Data Analysis. Wiley, New York ANSI/IEEE, 1991 Standard Glossary of Software Engineering Terminology, STD-729 ANSI/IEEE, 1990.
2. Akaike H. A new look at statistical model identification, IEEE Transactions on Automatic Control 1974; 19:716-723.
3. Anderson T, Lee P. Fault Tolerance: Principles and Practices, Prentice-Hall, Englewood Cliffs, 1980.
4. Anderson T, Barrett P, Halliwell D, Moulding M. “Software fault tolerance: An evaluation, *IEEE* Transactions on Software Engineering, 1985, 11(12).
5. Anderson T, Barrett P, Halliwell D, Moulding M. “Software fault tolerance: An evaluation, *IEEE* Transactions on Software Engineering, 1985, 11(12).
6. Tohma Y, Yamano H, Ohba M, Jacoby R. The estimation of parameters of the hypergeometric distribution and its application to the software reliability, 1991.
7. Growth model, *IEEE Trans.* Software Engineering, 17(5).
8. Tokuno K, Yamada S. Markovian availability measurement and assessment for hardware-software systems, Int. J Reliability, Quality and Safety Engineering, 1997, 4(3).
9. Voas JM, Miller KW. Software testability: The new verification, *IEEE Software* 1995; 12:17-28.
10. Wang H, Pham H. Optimal Age-Dependent Preventive Maintenance Policies with Imperfect Maintenance, International Journal of Reliability, Quality and Safety Engineering 1996; 3(2):119-135.
11. Subramanian GH, Breslawski S. An empirical analysis of software effort estimate alternations, Journals of Systems Software 1995; 31:135-141.
12. Schneberger SL. Distributed computing environments: Effects on software maintenance difficulty, *Journal of Systems Software* 1997; 37:101-116.
13. Randell B. System structure for software fault tolerance, IEEE Transactions on Software Engineering 1975; 1(2):220-232.
14. Hoang Pham, Handbook of Reliability Engineering, Springer: Apr, edition1, 2003.