Software Reliability Measuring using Modified Maximum Likelihood Estimation and SPC

Dr. R Satya Prasad
Associate Prof, Dept. of CSE
Acharya Nagarjuna University
Guntur, INDIA

K Ramchand H Rao
Dept. of CSE
A.S.N. Degree College
Tenali, INDIA

Dr. R.R. L Kantha
Professor, Dept. of Statistics
Acharya Nagarjuna University
Guntur, INDIA

ABSTRACT
Software reliability may be used as a measure of the Software system's success in providing its function properly. Software process improvement helps in finishing with reliable software product. Software process improvement includes monitoring software development practices and actively seeking ways to increase value, reduce errors, increase productivity, and enhance the developer's environment. Statistical process control (SPC) is one of the best available approaches to monitor and control the software process. SPC is the application of appropriate statistical tools to processes for continuous improvement in quality, reliability of software products and services and productivity in the workforce. In this paper we proposed a control mechanism, based on time between failures observations using Half logistic distribution, with Modified Maximum likelihood Estimation (MMLE) which is based on Non Homogenous Poisson Process (NHPP).

General Terms
Software Process, Control Charts, Software Quality.

Keywords
Modified MLE (MMLE), Half logistic Distribution (HLD), Statistical Process Control (SPC), Software reliability, Control limits, Non Homogenous Poisson Process (NHPP).

1. INTRODICTION
Software reliability is the most important and most measurable aspect of software quality and it is very customer oriented. It is a measure of how well the program functions to meet its operational requirements. Software reliability measures, guide the developer to better decisions. In the system engineering stage, they promote quantitative specification of design goals, schedules and resources required. The measures also help in the better management of project resources. The user will also benefit from software reliability measure, because the user is concerned with efficient operation of the system. If the operational needs with respect to quality are in accurately specified, the user will either get a system at an excessively high price or with an excessively high operational cost[3].

The most common approach to developing software reliability models is the probabilistic approach. The probabilistic model represents the failure occurrences and the fault removals as probabilistic events. There are numerous software reliability models available for use according to probabilistic assumptions. They are classified into various groups, including error seeding models, failure rate models, curve fitting models, reliability growth models, Markov structure models, and non-homogenous passion process (NHPP) models. The NHPP based models are the most important because of their simplicity, convenience and compatibility [6]. The monitoring of Software reliability process is a far from simple activity. In recent years, several authors have recommended the use of SPC for software process monitoring [10] [15]. Over the years, SPC has come to be widely used among others, in manufacturing industries for the purpose of controlling and improving processes. Our effort is to apply SPC techniques in the software development process so as to improve software reliability and quality [7] [15]. SPC is a method of process management through application of statistical analysis, which involves and includes the defining, measuring, controlling, and improving of the processes [9]. In measuring software reliability control charts can be used as efficient and appropriate SPC Tools [1][2]. The proposed process involves evaluation of the parameter of the Mean Value function and hence the values of the mean value function at various inter failure times to develop relevant time control chart. Satya Prasad et al [2011] [13] studied the same problem by evaluating the parameters with well classical maximum likelihood Estimation (MLE). This method involves numerical iterative solutions of equations, which sometimes create problems of convergence. To overcome this inconvenience we adopted a modification for the likelihood function in this paper. Details are provided in the following section.

2. MODEL FORMULATION
In order to overcome the numerical iterative way of solving the log likelihood equations and to get analytical estimators rather than iterative, some approximations in estimating the equations can be adopted from Kantam and Dharmarao (1994) [4] [12] , Kantam and Sriram (2001) [5] and the references there in. We use one such approximations here to get modified MLEs of 'a' and 'b'.

The simplified form of log likelihood equation is

$$b\sum_{k=1}^{n} s_k - n - 2\sum_{k=1}^{n}\frac{z_k.e^{-z_k}}{(1+e^{-z_k})} + \frac{2nz_k.e^{-z_k}}{(1-e^{-2z_k})} = 0 \quad (2.1)$$

Let us approximate the following expressions in the L.H.S of equation (2.1) by linear functions in the neighborhoods of the corresponding variables.

$$\frac{z_k.e^{-z_k}}{(1+e^{-z_k})} = \alpha_k + \beta_k.z_k, k = 1,2,3,\ldots.n \quad (2.2)$$

$$\frac{z_n.e^{-z_n}}{1-e^{-2z_n}} = \gamma_n + \delta_n.z_n \quad (2.3)$$

where the slopes α_k, γ_n and intercepts β_k, δ_n in equations (2.2) and (2.3) are to be suitably found. With such values equations (2.2) and (2.3) when used in equation (2.1) would give an approximate MLE for 'b' as

$$\hat{b} = \frac{n+2\sum_{k=1}^{n}\alpha_k - 2n\gamma_n}{\sum_{k=1}^{n}s_k - 2\sum_{k=1}^{n}\beta_k . s_k + 2n\delta_n s_n} \qquad (2.4)$$

We suggest following method to get the slopes and intercepts in the R.H.S of equations (2.2) and (2.3)

Let $F(z) = \left[\frac{1-e^{-z}}{1+e^{-z}}\right]$

$$\hat{p_i} = \frac{i}{n+1}, i = 1,2,3,\ldots n$$

$$F(\hat{u_i'}) = p_i - \sqrt{\frac{p_i(1-p_i)}{n}}$$

$$F(\hat{u_i''}) = p_i + \sqrt{\frac{p_i(1-p_i)}{n}}$$

Given a natural number 'n' we can get the values of $u_i' \text{ and } u_i''$ by inverting the above equations through the function F(z). If G (.), H (.) are the symbols for the L.H.S of equations (2.2) and (2.3) we get

$$\hat{\beta_k} = \frac{G(u_k'') - G(u_k')}{u_k'' - u_k'}, k = 1,2,3,4,\ldots n$$

$$\hat{\alpha_k} = G(u_k') - \beta_k . u_k', k = 1,2,3,4,\ldots n$$

$$\hat{\delta_n} = \frac{H(u_k'') - H(u_k')}{u_k'' - u_k'}$$

$$\hat{\gamma_n} = H(u_n') - \delta_n u_n'$$

Given the data observations and sample size using these values along with the sample data in equation (2.4) we get an approximate MLE of 'b'. Equation (3.4) gives approximate MLE of 'a'.

3. ESTIMATION BASED ON INTER FAILURE TIMES

The mean value function and intensity function of Half Logistic Model [6] are given by

$$m(t) = \frac{a(1-e^{-bt})}{(1+e^{-bt})}, a > 0, b > 0, t \geq 0 \qquad (3.1)$$

$$\lambda(t) = \frac{2abe^{-bt}}{(1+e^{-bt})^2} \qquad (3.2)$$

The constants 'a', 'b' which appear in the mean value function and hence in NHPP, in intensity function (error detection rate) and various other expressions are called parameters of the model. In order to have an assessment of the software reliability 'a',' b' are to be known or they are to be estimated from a software failure data.

Suppose we have 'n' time instants at which the first, second, third..., n[th] failures of a software are experienced. In other words if S_k is the total time to the k[th] failure, s_k is an observation of random variable S_k and 'n' such failures are successively recorded. The joint probability of such failure time realizations $s_1, s_2, s_3, \ldots . s_n$ is

$$L = e^{-m(s_n)} . \prod_{k=1}^{n} \lambda(s_n) \qquad (3.3)$$

The function given in equation (3.3) is called the likelihood function of the given failure data. Values of 'a', ' b' that would maximize L are called maximum likelihood estimators (MLEs) and the method is called maximum likelihood (ML) method of estimation. Accordingly 'a', 'b' would be solutions of the equations

$$\frac{\partial log L}{\partial a} = 0$$

$$\frac{\partial log L}{\partial b} = 0$$

$$\frac{\partial^2 log L}{\partial b^2} = 0$$

Substituting the expressions for m(t), λ(t) given by equations (3.1) and (3.2) in equation (3.3), taking logarithms, differentiating with respect to 'a', 'b' and equating to zero, after some joint simplification we get

$$a = n\left[\frac{1+e^{-bs_n}}{1-e^{-bs_n}}\right] \qquad (3.4)$$

$$g(b) = \sum_{k=1}^{n} s_k - \frac{n}{b} - 2\sum_{k=1}^{n-1} \frac{s_k e^{-bs_k}}{(1+e^{-bs_n})} - \frac{2s_n e^{-bs_k}}{(1+e^{-bs_n})}\left[1 - \frac{n}{1-e^{-bs_n}}\right] = 0 \qquad (3.5)$$

The value of 'b' can be obtained using Newton-Raphson method which when substituted in equation (3.4) gives value of 'a'.

4. MONITORING THE TIME BETWEEN FAILURES USING CONTROL CHART

The selection of proper SPC charts is essential to effective statistical process control implementation and use. There are many charts which use statistical techniques. It is important to use the best chart for the given data, situation and need[11].

There are advances charts that provide more effective statistical analysis. The basic types of advanced charts, depending on the type of data are the variable and attribute charts. Variable control chats are designed to control product or process parameters which are measured on a continuous

measurement scale. X-bar, R charts are variable control charts [14].

Attributes are characteristics of a process which are stated in terms of good are bad, accept or reject, etc. Attribute charts are not sensitive to variation in the process as variables charts. However, when dealing with attributes and used properly, especially by incorporating a real time pareto analysis, they can be effective improvement tools. For attribute data there are : p-charts, c-charts, np-charts, and u-charts. We have named the control chart as Failures Control Chart in this paper. The said control chart helps to assess the software failure phenomena on the basis of the given inter- failure time data[8].

4.1 Distribution of Time between failures

For a software system during normal operation, failures are random events caused by, for example, problem in design or analysis and in some cases insufficient testing of software. In this paper we applied *Half Logistic Distribution*[11] [12] to time between failures data. This distribution uses cumulative time between failure data for reliability monitoring.

The equation for mean value function of Half Logistic Distribution from equation 2.1

$$m(t) = a\left[\frac{1-e^{-bt}}{1+e^{-bt}}\right]$$

Equate the pdf of above m(t) to 0.99865, 0.00135, 0.5 and the respective control limits are given by.

$$m(t) = a\left[\frac{1-e^{-bt}}{1+e^{-bt}}\right] = 0.99865$$

It gives

$$t = \frac{7.300122639}{b} = t_U \qquad (4.1)$$

Similarly

$$t = \frac{0.002700002}{b} = t_L \qquad (4.2)$$

$$t = \frac{1.098612289}{b} = t_C \qquad (4.3)$$

The control limits are such that the point above the $m(t_U)$ (4.1)(UCL) is an alarm signal. A point below the $m(t_L)$(4.2) (LCL) is an indication of better quality of software. A point within the control limits indicates stable process.

4.2 Example

The procedure of a failures control chart for failure software process will be illustrated with an example here. Table 1 shows the time between failures of a software product [8].

Table -1: Cummulative Inter failures Time Data

Failure number	*Time between Failure (hrs) (cumulative)*	*Failure number*	*Time between Failure (hrs) (cumulative)*	*Failure number*	*Time between Failure (hrs) (cumulative)*
1	30.02	11	115.34	21	256.81
2	31.46	12	121.57	22	273.88
3	53.93	13	124.97	23	277.87
4	55.29	14	134.07	24	453.93
5	58.72	15	136.25	25	535
6	71.92	16	151.78	26	537.27
7	77.07	17	177.5	27	552.9
8	80.9	18	180.29	28	673.68
9	101.9	19	182.21	29	704.49
10	114.87	20	186.34	30	738.68

Table 2 shows the time between failures (cumulative) in hours, corresponding m(t) and successive difference between m(t)'s.

Table 2- Successive difference of mean value function (m(t))

Failure number	*Time between Failure (hrs) (cumulative)*	*m(t)*	*Successive Difference of m(t)*	*Failure number*	*Time between Failure (hrs) (cumulative)*	*m(t)*	*Successive Difference of m(t)*
1	30.02	2.029923497	0.097077791	16	151.78	9.923046372	1.537552672
2	31.46	2.127001289	1.508322141	17	177.5	11.46059904	0.163193461

3	53.93	3.63532343	0.090815888	18	180.29	11.62379251	0.111880954
4	55.29	3.726139318	0.228756749	19	182.21	11.73567346	0.239476794
5	58.72	3.954896066	0.876131706	20	186.34	11.97515025	3.81959287
6	71.92	4.831027772	0.339809004	21	256.81	15.79474312	0.844951713
7	77.07	5.170836776	0.251905868	22	273.88	16.63969484	0.192817725
8	80.9	5.422742644	1.367531191	23	277.87	16.83251256	6.758258675
9	101.9	6.790273835	0.831570421	24	453.93	23.59077124	2.071137895
10	114.87	7.621844256	0.029928296	25	535	25.66190913	0.050033654
11	115.34	7.651772553	0.395282189	26	537.27	25.71194279	0.333705373
12	121.57	8.047054742	0.214578039	27	552.9	26.04564816	2.021062108
13	124.97	8.26163278	0.57016458	28	673.68	28.06671027	0.382797878
14	134.07	8.831797361	0.135664868	29	704.49	28.44950815	0.373798464
15	136.25	8.967462229	0.955584143	30	738.68	28.82330661	--

The values of 'a' and 'b' are computed by using the well know iterative Newton-Rapson method. These values are used to compute, T_u, T_L, T_c i.e. UCL, LCL, CL. The values of a and b are 31.27686 and 0.00433 and

$m(T_U)/UCL = 31.23462967$

$m(T_L)/LCL = 0.042223764$

$m(T_C)/CL = 15.63842905$

The values of m(t) at T_c, T_u, T_L and at the given 30 inter-failure times are calculated. Then the m(t)'s are taken, which leads to 29 values. The graph with the said inter-failure times 1 to 30 on X-axis, the 29 values of m(t)'s on Y-axis, and the 3 control lines parallel to X-axis at $m(T_L)$, $m(T_U)$, $m(T_C)$ respectively constitutes failures control chart to assess the software failure phenomena on the basis of the given inter-failures time data.

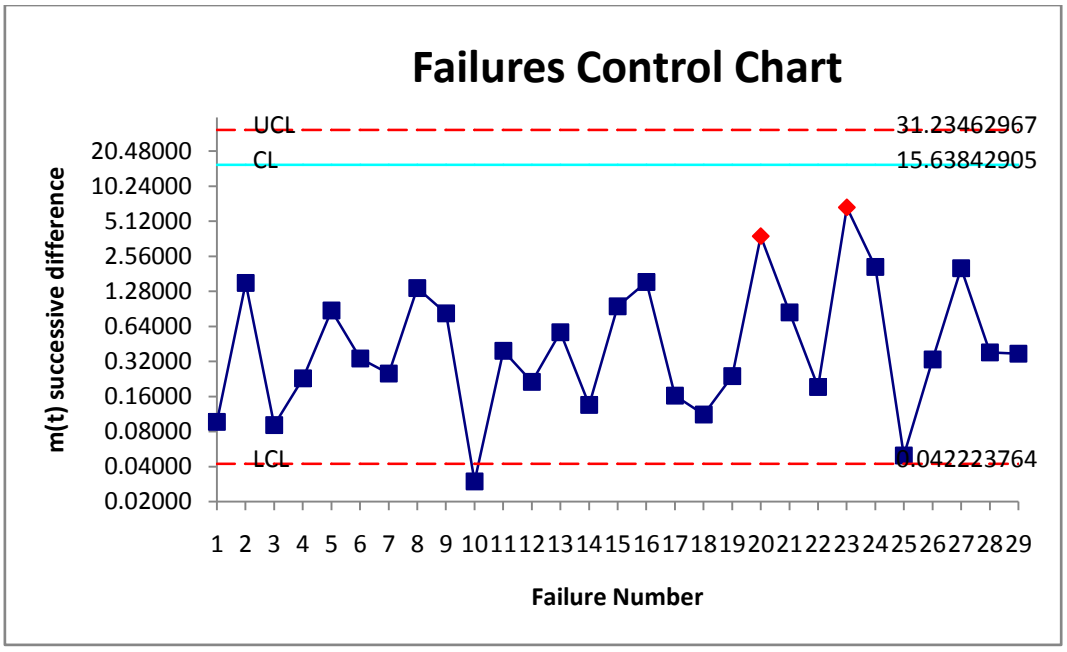

Fig 1

5. CONCULUSION

This failure control chart (Fig 1) exemplifies that, the first out of control situation is noticed at the 10^{th} failure with the corresponding successive difference of m(t) falling below the LCL. It results in an earlier and hence preferable out - of - control for the product. The assignable cause for this is to be investigated and promoted. The out of control signals in and the model suggested in Satya Prasad at el [2011] [13] are the same. We therefore conclude that adopting a modification to the likelihood method doesn't alter the situation, but simplified the procedure of getting the estimates of the parameters, thus resulting in a preference of the present model to the one described in Satya Prasad et al [2011] [13].

6. ACKNOWLEDGMENTS

Our thanks to Department of Computer Science and Engineering; Department of Statistics, Acharya Nagarjuna University; Department of Computer Sceince, Annabathuni Satyanaraya Degree College, Tenali, for providing necessary facilities to carryout the research work.

7. REFERENCES

[1] Burr,A. and Owen ,M.1996. Statistical Methods for Software quality . Thomson publishing Company. ISBN 1-85032-171-X.

[2] Carleton, A.D. and Florac, A.W. 1999. Statistically controlling the Software process. The 99 SEI Software Engineering Symposimn, Software Engineering Institute, Carnegie Mellon University.

[3] John D.Musa; "Software Quality and Reliability Basics"; AT&T Bell Laboratories. CH 2468-7/87/0000/014,1987 IEEE.

[4] Kantam, R.R.L;and Dharmarao V (1994) "Half Logistic Distribution - An improvement over M.L Estimator",proceedings of 11 Annual conference of SDS, 39-44.

[5] Kantam, R.R.L;and Sriram B (2001) "Variable Control Charts Based on Gama Distribution", IAPQR Transactions 26(2), 63-78.

[6] Khaled M.S. Faqih; "Proceedings of the International Multi Conference of Engineers and Computer Scientists 2009", Vol I IMECS 2009, March 18-20, Hongkong.

[7] K. U. Sargut, O. Demirors; Utilization of statistical process control (SPC) in emergent software organizations: Pitfallsand suggestions; Springer Science + Business media Inc. 2006.

[8]M.Xie, T.N. Goh, P. Rajan; Some effective control chart procedures for reliability monitoring; Elsevier science Ltd, Reliability Engineering and system safety 77(2002) 143- 150

[9]Mutsumi Komuro; Experiences of Applying SPC Techniques to software development processes; 2006 ACM 1-59593-085-x/06/0005.

[10] N. Boffoli, G. Bruno, D. Cavivano, G. Mastelloni; Statistical process control for Software: a systematic approach; 2008 ACM 978-1-595933-971-5/08/10.

[11] R.SatyaPrasad,Half Logistic Software Reliablity Growth Model,Ph.D. Thesis,2007

[12] R.SatyaPrasad, "Half Logistic Software Reliablity Growth Model",; International Journal of Advanced Research in Computer Sceince, Vol.1.No.2, Jul-Aug,2010. 31-36.

[13] R.SatyaPrasad, " Software Reliability with SPC"; International Journal of Computer Science and Emerging Technologies; Vol 2, issue 2, April 2011. 233-237.

[14] Ronald P.Anjard;SPC CHART selection process;Pergaman 0026-27(1995)00119-0 Elsevier science ltd.

[15] John Oakland; Statistical Process Control; Elsevier.