Discovery of Reliable Software using GOM on Interval Domain Data

Dr. R. Satya Prasad
Associate Professor
Dept. of CS & Engg.
Acharya Nagarjuna University

D. Haritha
Assistant Professor
Dept. of Electronics & Computer Engg.
K.L. University

ABSTRACT
Software reliability growth models using Non-Homogeneous Poisson Process(NHPP) with a mean value function –dependent on Linearly falling fault detection rate as proposed in the literature is considered. The well known Sequential Probability Ratio Test (SPRT) procedure of statistical science is adopted for the model in order to decide upon the reliability/ unreliability of developed software. The performance of the proposed model is demonstrated by using 6 Data Sets.

General Terms
Unreliable Software, exponential distribution, Mean value function, Intensity function , Statistical Quality Control..

Keywords
GOM, Maximum Likelihood Estimation, Decision lines, Software testing, Software failure data.

1. INTRODUCTION
In the analysis of software failure data we often deal with either inter failure times or number of recorded failures in a given time interval. If it is further assumed that the average number of recorded failures in a given time interval is directly proportional to the length of the interval and the random number of failure occurrences in the interval is explained by a Poisson process then we know that the probability equation of the stochastic process representing the failure occurrences is given by a homogeneous Poisson process with the expression

$$P\left[N(t)=n\right]=\frac{e^{-\lambda t}\left(\lambda t\right)^n}{n!} \qquad (1.1)$$

Stieber (1997) observes that if classical testing strategies are used (no usage testing), the application of software reliability growth models may be difficult and reliability predictions can be misleading. However, he observes that statistical methods can be successfully applied to the failure data. He demonstrated his observation by applying the well-known sequential probability ratio test (SPRT) of Wald (1947) for a software failure data to detect unreliable software components and compare the reliability of different software versions. In this paper we consider a popular SRGM – proposed by Goel and Okumoto(1979) and adopt the principle of Stieber (1997) in detecting unreliable software components in order to accept/reject a developed software. For brevity we denote the SRGM as GOM. The failure intensity is linearly decreasing in its mean value function. The theory proposed by Stieber (1997) is presented in Section 2 for a ready reference. Extension of this theory to the SRGM's – GOM is presented in Section 3. The procedure for parameter estimation is presented in section 4 Application of the decision rule to detect unreliable software components with respect to the proposed SRGM is given in Section 5.

2. WALD'S SEQUENTIAL TEST FOR A POISSON PROCESS
The sequential probability ratio test (SPRT) was developed by A.Wald at Columbia University in 1943. Due to its usefulness in development work on military and naval equipment it was classified as 'Restricted' by the Espionage Act (Wald, 1947). A big advantage of sequential tests is that they require fewer observations (time) on the average than fixed sample size tests. SPRTs are widely used for statistical quality control in manufacturing processes. An SPRT for homogeneous Poisson processes is described below.

Let $\{N(t),t \geq 0\}$ be a homogeneous Poisson process with rate 'λ'. In our case, N(t)=number of failures up to time ' t' and ' λ' is the failure rate (failures per unit time). Suppose that we put a system on test (for example a software system, where testing is done according to a usage profile and no faults are corrected) and that we want to estimate its failure rate ' λ'. We cannot expect to estimate ' λ' precisely. But we want to reject the system with a high probability if our data suggest that the failure rate is larger than λ_1 and accept it with a high probability, if it's smaller thanλ_0 $(0 < \lambda_0 < \lambda_1)$. As always with statistical tests, there is some risk to get the wrong answers. So we have to specify two (small) numbers 'α' and 'β', where 'α' is the probability of falsely rejecting the system. That is rejecting the system even if $\lambda \leq \lambda_0$. This is the "producer's" risk. β is the probability of falsely accepting the system .That is accepting the system even if $\lambda \geq \lambda_1$. This is the "consumer's" risk. With specified choices of λ_0 and λ_1 such that $0 < \lambda_0 < \lambda_1$, the probability of finding N(t) failures in the time span (0,t) with λ_1,λ_0 as the failure rates are respectively given by

$$P_1=\frac{e^{-\lambda_1 t}\left[\lambda_1 t\right]^{N(t)}}{N(t)!} \qquad (2.1)$$

$$P_0 = \frac{e^{-\lambda_0 t}\left[\lambda_0 t\right]^{N(t)}}{N(t)!} \qquad (2.2)$$

The ratio $\frac{P_1}{P_0}$ at any time't' is considered as a measure of deciding the truth towards λ_0 or λ_1, given a sequence of time instants say $t_1 < t_2 < t_3 < .... < t_K$ and the corresponding realizations. $N(t_1), N(t_2), ... N(t_K)$ of N(t). Simplification of $\frac{P_1}{P_0}$ gives

$$\frac{P_1}{P_0} = \exp(\lambda_0 - \lambda_1)t + \left(\frac{\lambda_1}{\lambda_0}\right)^{N(t)}$$

The decision rule of SPRT is to decide in favor of λ_1, in favor of λ_0 or to continue by observing the number of failures at a later time than 't' according as $\frac{P_1}{P_0}$ is greater than or equal to a constant say A, less than or equal to a constant say B or in between the constants A and B. That is, we decide the given software product as unreliable, reliable or continue the test process with one more observation in failure data, according as

$$\frac{P_1}{P_0} \geq A \qquad (2.3)$$

$$\frac{P_1}{P_0} \leq B \qquad (2.4)$$

$$B < \frac{P_1}{P_0} < A \qquad (2.5)$$

The approximate values of the constants A and B are taken as

$$A \cong \frac{1-\beta}{\alpha}, \quad B \cong \frac{\beta}{1-\alpha}$$

Where 'α' and 'β' are the risk probabilities as defined earlier. A simplified version of the above decision processes is to reject the system as unreliable if N(t) falls for the first time above the line

$$N_U(t) = a.t + b_2 \qquad (2.6)$$

to accept the system to be reliable if N(t) falls for the first time below the line

$$N_L(t) = a.t - b_1 \qquad (2.7)$$

To continue the test with one more observation on (t, N(t)) as the random graph of [t, N(t)] is between the two linear boundaries given by equations (2.6) and (2.7) where

$$a = \frac{\lambda_1 - \lambda_0}{\log\left(\frac{\lambda_1}{\lambda_0}\right)} \qquad (2.8)$$

$$b_1 = \frac{\log\left[\frac{1-\alpha}{\beta}\right]}{\log\left(\frac{\lambda_1}{\lambda_0}\right)} \qquad (2.9)$$

$$b_2 = \frac{\log\left[\frac{1-\beta}{\alpha}\right]}{\log\left(\frac{\lambda_1}{\lambda_0}\right)} \qquad (2.10)$$

The parameters α, β, λ_0 and λ_1 can be chosen in several ways. One way suggested by Stieber (1997) is

$$\lambda_0 = \frac{\lambda.\log(q)}{q-1}, \; \lambda_1 = q\frac{\lambda.\log(q)}{q-1}$$

$$where\ q = \frac{\lambda_1}{\lambda_0}$$

If λ_0 and λ_1 are chosen in this way, the slope of N_U (t) and N_L (t) equals λ. The other two ways of choosing λ_0 and λ_1 are from past projects (for a comparison of the projects) and from part of the data to compare the reliability of different functional areas (components).

3. SEQUENTIAL TEST FOR SOFTWARE RELIABILITY GROWTH MODEL

In Section 2, for the Poisson process we know that the expected value of N(t) = λt called the average number of failures experienced in time 't' .This is also called the mean value function of the Poisson process. On the other hand if we consider a Poisson process with a general function (not necessarily linear) m(t) as its mean value function the probability equation of a such a process is

$$P\left[N(t) = Y\right] = \frac{\left[m(t)\right]^y}{y!}.e^{-m(t)}, y = 0,1,2,----$$

Depending on the forms of m(t) we get various Poisson processes called NHPP for our model the mean value function

$m(t) = a(1 - e^{-bt}) \quad where \quad a > 0, b > 0, t > 0$

We may write

$$P_1 = \frac{e^{-m_1(t)}.\left[m_1(t)\right]^{N(t)}}{N(t)!}$$

$$P_0 = \frac{e^{-m_0(t)}.\left[m_0(t)\right]^{N(t)}}{N(t)!}$$

where $m_1(t)$, $m_0(t)$ are values of the mean value function at specified sets of its parameters indicating reliable software and unreliable software respectively. For instance the model we have been considering its m(t) function, contains a pair of parameters a, b with 'a' as a multiplier. Also a, b are positive. Let P_0 , P_1 be values of the NHPP at two specifications of b say $b_0, b_1\left(b_0 < b_1\right)$ respectively. It can be shown that for our models m(t) at b_1 is greater than that at b_0. Symbolically $m_0(t)<m_1(t)$.Then the SPRT procedure is as follows:

Accept the system to be reliable $\frac{P_1}{P_0} \le B$

i.e., $\frac{e^{-m_1(t)}.\left[m_1(t)\right]^{N(t)}}{e^{-m_0(t)}.\left[m_0(t)\right]^{N(t)}} \le B$

i.e., $N(t) \le \frac{\log\left(\frac{\beta}{1-\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)}$ (3.1)

Decide the system to be unreliable and reject if $\frac{P_1}{P_0} \ge A$

i.e., $N(t) \ge \frac{\log\left(\frac{1-\beta}{\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)}$ (3.2)

Continue the test procedure as long as

$$\frac{\log\left(\frac{\beta}{1-\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)} < N(t) < \frac{\log\left(\frac{1-\beta}{\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)}$$

------------(3.3)

Substituting the appropriate expressions of the mean value function – m(t) of GOM we get the decision rules and are given in followings lines

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

Acceptance region:

$$N(t) \le \frac{\log\left(\frac{\beta}{1-\alpha}\right) + a\left(e^{-b_0 t} - e^{-b_1 t}\right)}{\log\left(\frac{1-e^{-b_1 t}}{1-e^{-b_0 t}}\right)} \quad (3.4)$$

Rejection region:

$$N(t) \ge \frac{\log\left(\frac{1-\beta}{\alpha}\right) + a\left(e^{-b_0 t} - e^{-b_1 t}\right)}{\log\left(\frac{1-e^{-b_1 t}}{1-e^{-b_0 t}}\right)} \quad (3.5)$$

Continuation region:

$$\frac{\log\left(\frac{\beta}{1-\alpha}\right) + a\left(e^{-b_0 t} - e^{-b_1 t}\right)}{\log\left(\frac{1-e^{-b_1 t}}{1-e^{-b_0 t}}\right)} < N(t) < \frac{\log\left(\frac{1-\beta}{\alpha}\right) + a\left(e^{-b_0 t} - e^{-b_1 t}\right)}{\log\left(\frac{1-e^{-b_1 t}}{1-e^{-b_0 t}}\right)} \quad (3.6)$$

It may be noted that in the above model the decision rules are exclusively based on the strength of the sequential procedure (α,β) and the values of the mean value functions namely, $m_0(t)$, $m_1(t)$. If the mean value function is linear in 't' passing through origin, that is, m(t) = λt the decision rules become decision lines as described by Stieber (1997). In that sense equations (3.1), (3.2) , (3.3) can be regarded as generalizations to the decision procedure of Stieber (1997).The applications of these results for live software failure data are presented with analysis in Section 4.

4. PARAMETER ESTIMATION

Parameter estimation is of primary importance in software reliability prediction. Once the analytical solution for m(t) is known for a given model, parameter estimation is achieved by applying a well known technique of Maximum Likelihood Estimation (MLE). 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, time domain data and interval domain data.

The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability (likelihood) of the sample data. The method of maximum likelihood is considered to be more robust (with some exceptions) and yields estimators with good statistical properties. In other words, MLE methods are versatile and apply to most models and to different types of data. Although the methodology for maximum likelihood estimation is simple, the implementation is mathematically intense.

Assuming that the data are given for the cumulative number of detected errors yi in a given time-interval $(0,t_i)$ where $i = 1,2, \ldots, n$. and $0 < t_1 < t_2 <\ldots< t_n$ then the log likelihood function (LLF) takes on the following form. Likely hood function by using λ(t) is: $L = \prod_{i=1}^{n} \lambda(t_i)$ .The logarithmic likelihood function for interval domain data (pham, 2006) is given by:

$$Log\ L = \sum_{i=1}^{n} (y_i - y_{i-1}) \log[m(t_i) - m(t_{i-1})] - m(t_n)$$

The maximum likelihood estimators (MLE) of $\theta_1, \theta_2, \ldots, \theta_k$ are obtained by maximizing L or Λ, where Λ is ln L . By maximizing Λ, which is much easier to work with than L, the maximum likelihood estimators (MLE) of $\theta_1, \theta_2, \ldots, \theta_k$ are the simultaneous solutions of k equations such that:

$$\frac{\partial(\wedge\Lambda)}{\partial\theta_j} = 0 \quad j=1,2,\ldots,k$$

The parameters 'a' and 'b' are estimated using iterative Newton Raphson Method, which is given as $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

To estimate 'a' and 'b' , for a sample of n units, first obtain the likelihood function: $L = \prod_{i=1}^{n} abe^{-bt}$ Take the natural logarithm on both sides, The Log Likelihood function is given as: $Log\ L = \log[\prod_{i=1}^{n} \lambda(t_i)]$

$$= \log[\prod_{i=1}^{n} abe^{-bt}]$$

$$= \sum_{i=1}^{n} (y_i - y_{i-1}) \log[a(1 - e^{-bt_i})\text{-}a(1\text{-}e^{-bt_{i-1}})]\text{-}a(1\text{-}e^{-bt_n})$$

The parameter 'a' is estimated by taking the partial derivative w.r.t 'a' and equating to '0'. (i.e $\frac{\partial \log L}{\partial a} = 0$)

$$a = \frac{y_n - y_0}{\left(1 - e^{-bt_n}\right)}$$

The parameter 'b' is estimated by iterative Newton Raphson Method using $b_{n+1} = b_n - \frac{g(b_n)}{g^{|}(b_n)}$. which is substituted in finding 'a'. where g(b) & $g^{|}$(b) are expressed as follows.

$$g(b) = \frac{\partial \log L}{\partial b} = 0 \quad ; \quad g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0$$

$$g(b) = \sum_{i=1}^{n} (y_i - y_{i-1}) \frac{\left(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}}\right)}{e^{-bt_{i-1}} - e^{-bt_i}} - \frac{(y_n - y_0) t_n e^{-bt_n}}{\left(1 - e^{-bt_n}\right)}$$

$$g'(b) = \frac{(y_n - y_0) t_n^2 e^{-bt_n}}{\left(1 - e^{-bt_n}\right)^2} - \sum_{i=1}^{n} (y_i - y_{i-1}) \frac{e^{-bt_i} e^{-bt_{i-1}} (t_i - t_{i-1})^2}{\left(e^{-bt_{i-1}} - e^{-bt_i}\right)^2}$$

5. SPRT ANALYSIS OF LIVE DATA SETS

We see that the developed SPRT methodology is for a software failure data which is of the form [t, N(t)] where N(t) is the observed number of failures of software system or its sub system in 't' units of time. In this section we evaluate the decision rules based on the considered mean value functions for six different data sets of the above form, borrowed from Wood (1996), Pham (2005). Based on the estimates of the parameter 'b' in each mean value function, we have chosen the specifications of b_0 , b_1 equidistant on either side of estimate of b obtained through a Data Set to apply SPRT such that $b_0 < b < b_1$. The choices are given in the following table.

Table 5.1: Specifications of b0, b1

Data Set	Estimate of a	Estimate of b	b_0	b_1
Pham (2005) Phase 1 Data	90.49	0.000044	0.000022	0.000066
Pham (2005) Phase 2 Data	116.89	0.000052	0.000030	0.000074
Wood (1996) Release 1 Data	122.31	0.000171	0.000149	0.000193
Wood (1996) Release 2 Data	199.44	0.000090	0.000068	0.000112
Wood (1996) Release 3 Data	79.17	0.000291	0.000269	0.000313
Wood (1996) Release 4 Data	132.36	0.000035	0.000013	0.000057

Using the selected b_0,b_1 and subsequently the $m_0(t)$,$m_1(t)$for each model we calculated the decision rules given by Equations 3.4, 3.5, sequentially at each 't' of the data sets taking the strength (α, β) as (0.05,0.05). These are presented for the model in Tables 5.2.

Table 5.2: SPRT for GOM

Data Set	T	N(t)	R.H.S of equation (3.4) Acceptance region (≤)	R.H.S of Equation (3.5) Rejection Region(≥)
Pham(2005) Phase 1 Data	356	1	-1.4201	3.9786
	712	1	-0.1817	5.5557
	1068	2	1.0352	6.5117
	1424	3	2.2311	7.7470
	1780	5	3.4064	8.9621
	2136	5	4.5615	10.1574
	2492	5	5.6967	11.3331
Pham (2005) Phase 2 Data				5.6373
	416	3	-0.9517	7.9606
	832	4	1.3043	10.2326
	1248	4	3.5079	12.4544
	1664	7	5.6605	14.6273
	2080	9	7.7632	16.7525
	2496	9	9.8174	
Wood (1996) Release 1 Data	529	16	-1.5693	22.2247
	968	24	6.1790	30.9185
	1430	27	13.489	39.2539
	1893	33	20.1892	47.0373
	2490	41	27.9742	56.3080
	3058	49	34.5605	64.4072
	3625	54	40.4048	71.8659
	4422	58	47.4984	81.4199
	5218	69	53.4035	90.0274
	5823	75	57.1792	96.0370
	6539	81	60.9222	102.6446
	7083	86	63.2790	107.3511
	7487	90	64.7738	110.6952
	7846	93	65.9278	113.5714
	8205	96	66.9236	116.3676
	8564	98	67.7667	119.0929
	8923	99	68.4615	121.7559
	9282	100	69.0123	124.3650
	9641	100	69.4228	126.9284
	10000	100	69.6962	129.4539
Wood (1996) Release 2 Data	384	13	0.6337	12.6373
Wood (1996) Release 3 Data	162	6	-16.2619	23.5397
	499	9	-10.2353	31.5953
	715	12	-6.7590	36.4441
	1137	20	-0.7730	45.2862
	1799	28	6.6721	57.7238
	2438	40	11.8653	68.4103
	2818	48	14.1215	74.2889
	3574	54	16.9382	85.2132
	4234	57	17.7178	94.1947
	4680	59	17.4118	100.1137
	4955	60	16.8984	103.7430
	5053	61	16.6561	105.0369
Wood (1996) Release 4 Data	254	1	-1.0039	2.9953
	788	3	1.0406	5.0718
	1054	8	2.0427	6.0898

From the above table we see that a decision either to accept or reject the system is reached much in advance of the last time instant of the data(the testing time).The following consolidated table reveals the iterations required to come to a decision about the software of each Data Set.

Table 5.3: Consolidated Table of Decisions

Data Set	GOM Model	Iterations	Decision
Pham (2005) Phase 1 Data	0.9752	7	Accept
Pham (2005) Phase 2 Data	0.9961	6	Accept
Wood (1996) Release 1 Data	0.9742	21	Continuous
Wood (1996) Release 2 Data	0.9742	1	Reject
Wood (1996) Release 3 Data	0.9290	13	Continuous
Wood (1996) Release 4 Data	0.9672	3	Reject

The above consolidated table shows that GOM as exemplified for 6 Data Sets indicate that the model is performing well for 2 Data Sets in arriving at a decision. For the remaining 4 Data Sets GOM has given decision for 2 Data Sets and inconclusive for 2 Data Sets. Therefore, we may conclude that the model GOM has an edge to arrive at a decision in deciding upon reliability / unreliability of software.

The authors are exploring the possibility of performance a new SRGM generated on the basis of dependence of mean value function on the fault detection rate in a quadratically decreasing manner.

REFERENCES

[1] GOEL, A.L and OKUMOTO, K. (1979). “A Time Dependent Error Detection Rate Model For Software Reliability And Other Performance Measures”, IEEE Transactions on Reliability, vol.R-28, pp.206-211, 1979.

[2] MUSA, J.D., and OKUMOTO, K. (1984). ”A Logorithmic Poisson Execution Time Model For Software Reliability Measurement”, Proceeding Seventh International Conference on Software Engineering, Orlando, 230-238.

[3] PHAM, H.(2005). ”A Generalized Logistic Software Reliability Growth Model”, OPSEARCH, Vol.42, No.4, 322-331.

[4] Pham. H., 2006. “System software reliability”, Springer.

[5] STIEBER, H.A.(1997). “Statistical Quality Control: How To Detect Unreliable Software Components”, Proceedings the 8th International Symposium on Software Reliability Engineering, 8-12.

[6] WALD (1947).”Sequential Analysis”, Wiley,New York.

[7] WOOD, A.(1996).“Predicting Software Reliability”, IEEE Computer, 2253-2264.

[8] R.Satya Prasad and G. Krishna Mohan. (2011). ”Detection Of Reliable Software Using SPRT On Time Domain Data”, International Journal of Computer Science, Engineering and Applications, Vol.1, No.4, pp.92-99.

[9] R. Satya Prasad, N. Supriya and G. Krishna Mohan (2011).“Detection Of Reliable Software Using SPRT”

[10] International Journal of Advanced Computer Science and Applications Vol.2, No: 8, pp.60-63.

[11] R. Satya Prasad and Krishna Mohan. G(2011). “Detection Of Reliable Software Using SPRT On Interval Domain Data” , Computer Engineering & Intelligent Systems Vol.2, No: 3, pp.86-93.

[12] R. Satya Prasad and D. Haritha (2011) . “ Detection of Reliable Software Using HLSRGM”, International Journal of Computer Information Systems, Vol. 3 , No. 2,pp.49-53.