IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 14, Issue 6 (Sep. - Oct. 2013), PP 37-42
www.iosrjournals.org

An SPRT Procedure for an Ungrouped Data using MMLE Approach

V. Goutham[1], R. Satya Prasad [2]
[1](*Department of Computer Science and Engineering,StMary College of Engineering / JNTUH, India)*
[2](*Department of Computer Science and Engineering, Acharya Nagrjuna University, Nagarjuna Nagar, Guntur, Andhrapradesh, India)*

Abstract: Sequential Analysis of statistical science could be adopted in order to decide up on the reliable/unreliable of the developed software very quickly. Sequential analysis is different from classical hypothesis testing. In classing hypothesis testing one can draw conclusion during the data collection and final conclusion can possible be reached at a later stage, whereas sequential analysis are easy to see as data collection can be terminated after fewer cases and decision taken earlier stage, thus saving in terms of reliability of software. In this paper we proposed the performance of sequential probability ratio test (SPRT) on time domain data using exponential model and analyzed the result by applying on 3 data sets. The parameters are estimated using Modified Maximum likelihood estimation.

Keywords: Software reliability, software failure data, Sequential Probability Ratio Test, Modified Maximum Likelihood Estimation, Mean Value Function.

I. INTRODUCTION

The sequential probability ratio test (SPRT) is a specific sequential hypothesis test, developed by Abraham Wald. Neyman and Pearson's 1933 result inspired Wald to reformulate it as a sequential analysis problem. Software reliability is the most important and most measurable aspect of software quality and it is very customer oriented. 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.

In Classical Hypothesis Testing, the data collection is executed without analysis and consideration of the data. After all the data is collected the analysis is done, conclusions are drawn where as sequential analysis is a method of statistical inference whose characteristic features is that number of observation required by the procedure is not determined in advance of the experiment. The decision to terminate the experiment depends, at each stage, on the results of the observation previously made. A merit of sequential method, as applied to testing statically hypothesis, is that test procedure can be constructed which require, on the average ,a substantially smaller number of observation that equally reliable test procedure based on a predetermined number of observations[1] [3]. This paper describes a method for detecting software faults based on the Sequential Probability Ratio Test (SPRT) using MMLE parameter estimation. The SPRT is the optimal statistical test that makes the correct decision in the shortest time among all tests that are subject to the same level of decision errors. As a result, it can be expected that the proposed method has the potential of providing the quickest detection of a fault compared with other methods with the same false-alarm and miss-detection rates. SPRT is issued to detect the fault based on the calculated likelihoods of the hypotheses.

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

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

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 (see [4][5][7]).

II. Wald's Sequential Test for a Poisson Process

The sequential probability ratio test 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).In statistical terms, software system fault detection is essentially a binary decision or hypothesis testing problem, it is either a fault or no fault. Statistical theory and methods for such a problem are fairly complete and mature. Statistical tests are available that are simple and optimal under various criteria.

All methods for hypothesis test may be sequential or nonsequential. In a nonsequential method, a fixed size of samples (i.e., a fixed number of measurements) is used and the decision is made based on this whole block of samples altogether. A sequential method uses the samples one by one and the decision may be made at any time when sufficient evidence is gathered. A sequential test consists of a stopping rule, which determines when the test is done, and a decision rule, determines which hypothesis to choose.

In software system computer relaying, sequential methods should be preferred primarily for the following reasons: The measurements are obtained sequentially sequential tests are usually substantially more efficient in terms of the use of information in the measurements than nonsequential tests, and thus lead to a quicker decision at the same level of decision errors, a sequential test does not need to determine the number of measurements for the test in advance, while a non-sequential test does. It is due to this last reason that some adhoc and questionable tricks to be adopted in the non-sequential hypothesis testing based technique to handle the case in which one sample is not enough to make a decision [6]. As a sequential method, SPRT has an additional advantage that its appropriate threshold for the test statistics can be determined easily without knowledge of the distribution of the measurements, while for a non-sequential test the threshold usually depends on the distribution. 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.

Let {N(t),t ≥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 than λ 0 (0 < λ 0 < λ 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 λ ≤λ0. This is the "producer's" risk. β is the probability of falsely accepting the system .That is accepting the system even if λ ≥ λ 1. This is the "consumer's" risk. With specified choices of λ 0 and λ 1 such that 0 < λ 0 < λ 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

$$P1 = \frac{[\lambda 1 t]^{N(t)} e^{-\lambda t}}{N(t)!} \quad \text{----------->(2.1)}$$

$$P0 = \frac{[\lambda 0 t]^{N(t)} e^{-\lambda t}}{N(t)!} \quad \text{----------->(2.2)}$$

The ratio $\frac{P1}{P0}$ at any time't' is considered as a measure of deciding the truth towards λ0 or λ1, given a sequence of time instants say t1<t2<t3 and the corresponding realizations $N(t_1), N(t_2), \ldots\ldots N(t_K)$ of N(t). Simplification of $\frac{P1}{P0}$ gives

$$\frac{P1}{P0} = \exp(\lambda 0 - \lambda 1)\, t + \left[\frac{\lambda 1}{\lambda 0}\right]^{N(t)}$$

The decision rule of SPRT is to decide in favor of λ_1, in favour of λ_0 or to continue by observing the number of failures at a later time than 't' according as P1/P0 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

$$\frac{P1}{P0} \geq A \quad \text{----------(2.3)}$$

$$\frac{P1}{P0} \geq B \quad \text{---------- (2.4)}$$

$$B < \frac{P1}{P0} < A \quad \text{---------- (2.5)}$$

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

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

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.

$$N_U(t) = a.t + b2 \quad \text{----------(2.6)}$$

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

$$N_L(t) = a.t - b2 \quad \text{----------(2.7)}$$

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

$$a = \frac{\lambda 1 - \lambda 0}{\log(\frac{\lambda 1}{\lambda 0})} \quad \text{----------(2.8)}$$

$$b1 = \frac{\log[(1-\alpha)/\beta]}{\log[\frac{\lambda 1}{\lambda 0}]} \quad \text{----------(2.9)}$$

$$b2 = \frac{\log[(1-\beta)/\alpha]}{\log[\frac{\lambda 1}{\lambda 0}]} \quad \text{----------(2.10)}$$

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

$$\lambda 0 = \frac{\lambda \log(q)}{q-1} \quad , \lambda 1 = q.\frac{\lambda \log(q)}{q-1}$$

$$\text{where } q = \frac{\lambda 1}{\lambda 0}$$

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).

III. Sequential Test for Software Reliability Growth Models

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

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

Depending on the forms of m(t) we get various Poisson processes called NHPP[11]. For our model the mean value function is given as $m(t)=a(1-e^{-bt})$ where a>0,b>0,t>0

We may write

$$P1 = \frac{[m1(t)]^{N(t)} e^{-m1(t)}}{N(t)!}$$

$$P0 = \frac{[m0(t)]^{N(t)} e^{-m0(t)}}{N(t)!}$$

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 where (b0< *b*1 *)* 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 $\frac{P1}{P0} \leq B$

$$\text{ie: } \frac{e^{-m1(t)}\ [m1(t)]^{N(t)}}{e^{-m0(t)}\ [m0(t)]^{N(t)}} \leq B$$

$$\text{ie: } N(t) \leq \frac{\log\left(\frac{\beta}{1-\alpha}\right)+m1(t)-m0(t)}{logm1(t)-logm0(t)} \quad \text{---------- (3.1)}$$

Decide the system to be unreliable and reject if $\frac{P1}{P0} \geq A$

$$\text{ie: } N(t) \geq \frac{\log\left(\frac{1-\beta}{\alpha}\right)+m1(t)-m0(t)}{logm1(t)-logm0(t)} \quad \text{---------- (3.2)}$$

Continue the test procedure as long as

$$\frac{\log\left(\frac{1-\beta}{\alpha}\right)+m1(t)-m0(t)}{logm1(t)-logm0(t)} \leq N(t) \leq \frac{\log\left(\frac{\beta}{1-\alpha}\right)+m1(t)-m0(t)}{logm1(t)-logm0(t)} \quad \text{----------(3.3)}$$

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

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

Acceptance region:

$$N(t) \leq \frac{\log\left(\frac{\beta}{1-\alpha}\right)+m1(t)-m0(t)}{logm1(t)-logm0(t)}$$

$$N(t) \leq \frac{\log\left(\frac{\beta}{1-\alpha}\right)+a(e^{-b0t}-e^{-b1t})}{\log\left[\frac{1-e^{-b1t}}{1-e^{-b0t}}\right]} \quad \text{----------(3.4)}$$

Rejection region:

$$N(t) \geq \frac{\log\left(\frac{1-\beta}{\alpha}\right)+a(e^{-b0t}-e^{-b1t})}{\log\left[\frac{1-e^{-b1t}}{1-e^{-b0t}}\right]} \quad \text{----------(3.5)}$$

Continuation region:

$$\frac{\log\left(\frac{\beta}{1-\alpha}\right)+a(e^{-b0t}-e^{-b1t})}{\log\left[\frac{1-e^{-b1t}}{1-e^{-b0t}}\right]} < N(t) < \frac{\log\left(\frac{1-\beta}{\alpha}\right)+a(e^{-b0t}-e^{-b1t})}{\log\left[\frac{1-e^{-b1t}}{1-e^{-b0t}}\right]} \quad \text{----------(3.6)}$$

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 respective 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 decision rules become decision lines as described by Stieber (1997). In that case 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 IV. In the below picture the horizontal axis has been normalized. After normalizing the time axis the same graph can be used for different failure rates.The equations of lines only depends on α ,β and $\frac{\lambda 1}{\lambda 0}$ (see [2] and [4].

IV. Modified Maximum Likelihood Estimation (MMLE)

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 modified maximum likelihood estimation. An analytical approximation is used instead of linear approximation for a function which appears in Maximum Likelihood equation. These estimates are shown to perform better, in the sense of simplicity of calculation than the one based on linear approximation for the same function. In this paper we identified a Modified Maximum Likelihood Estimation (MMLE) based scheme to estimate software reliability using exponential distribution [12] [5].

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

$$L = e^{-m(s_n).} \prod_{k=1}^{n} \lambda(s_k) \qquad \text{----------(4.1)}$$

The simplified form for log likelihood equation of Exponential Distribution is [1]

$$\sum_{k=1}^{n} S_k a^{n-k} \frac{n}{b} - ans_n \; e^{-bs_n} = 0 \qquad \text{----------(4.2)}$$

Let us approximate the following expressions in the L.H.S of equation (4.2) by linear functions in the neighborhoods of the corresponding variables.

$$\frac{s_n e^{-bs_n}}{1-e^{-bs_n}} = mS_n + c \text{ , } n = 1, 2, \ldots\ldots n. \qquad \text{----------(4.3)}$$

$S_n$is the slope and 'c' is the intercepts in equations (4.3) which are suitably found. With such values equations (4.3) when used in equation (4.2) would give an approximate MLE for 'b' as

$$\hat{b} = 1 + \frac{1}{(mS_n + c) + \bar{s}} \qquad \text{----------(4.4)}$$

$$\text{where } \bar{s} = \sum_{k=1}^{n} \frac{S_k}{n}$$

We suggest the following method to get the slopes and intercepts in the R.H.S of equations (4.6) (4.7).

$$F(z) = \frac{n}{n+1} \qquad \text{----------(4.5)}$$

$$F(Z') = p - \sqrt{\frac{pq}{n}} \qquad \text{----------(4.6)}$$

$$F(Z'') = p + \sqrt{\frac{pq}{n}} \qquad \text{----------(4.7)}$$

Given a natural number 'n' we can get the values of Z' and Z'' by inverting the above equations through the function F(z) the L.H.S of equations (4.3) we get

$$m = \frac{\left(\frac{z'e^{-z'}}{1-e^{-z'}}\right) - \left(\frac{z''e^{-z''}}{1-e^{-z''}}\right)}{z' - z''} \qquad \text{----------(4.8)}$$

$$c = \frac{z'e^{-z'}}{1-e^{-z'}} - m\,z' \qquad \text{----------(4.9)}$$

It can be seen that the evaluation of S_n , C are based on only a specified natural number 'n' and can be computed free from any data. Given the data observations and sample size using these values along with the sample data in equation (4.8)(4.9) we get an approximate MLE of 'b'. Equation (4.10)gives approximate MLE of 'a'.

$$a = \frac{n}{1-e^{-bsn}} \qquad \text{---------- (4.10)}$$

V. Numerical Analysis Of Data Sets Using SPRT

Based on the time between failures data give in Table-1, we compute the two unknown parameters of a and b .

Table -1: Parameter Estimation using Inter Failure Times Data:

Data Set	Estimation of a	Estimation of b	bo	b1
DS1	31.6982	0.00396	0.00146	0.00646
DS2	17.2306314	0.006907	0.004407	0.009407
DS3	29.71851	0.008314	0.005814	0.010814

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 failure number 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 function for Five different data sets of the above form, borrowed from [8][9][10] with the assumption of c=0.05. Based on the estimates of the parameter 'b' in each mean value function, we have chosen the specifications of b0=b-δ, b1=b+δ, equidistant on either side of estimate of b obtained through a data set to apply SPRT such that b0 < b < b1. Assuming the value of δ=0.0025, the choices are given in the following table.

Using the selected b0 and b1 subsequently the $m_0(t)$,$m_1(t)$ for the 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, 0.2). These are presented for the model in Table II.

TABLE II. SPRT ANALYSIS FOR 3 DATA SETS:

Data Set	T	N(t)	R.H.S of equation (3.4) Acceptance region (≤)	R.H.S of Equation (3.5) Rejection Region(≥)	Decision
DS 1	**30.2**	**1**	1.889062	4.952618	**Accept**
DS 2	**10**	**1**	-1.027918	4.875902	
	19	**2**	-0.171415	5.912798	
	32	**3**	0.960673	7.318140	
	43	**4**	1.828804	8.429804	
	58	**5**	2.891387	9.843786	**Continue**
	70	**6**	3.649045	10.899561	
	88	**7**	4.646110	12.374139	
	103	**8**	5.360976	13.516708	
	125	**9**	6.239173	15.075428	
	150	**10**	7.019353	16.713339	
	169	**11**	7.475009	17.887203	
	199	**12**	7.980780	19.657385	
	231	**13**	8.263060	21.487215	
	256	**14**	8.316586	22.912397	
	296	**15**	8.120652	25.254193	
DS 3	**9**	**1**	-0.529991	6.707761	
	21	**2**	1.867279	9.470397	**Accept**
	32	**3**	3.841018	11.800234	

VI. Conclusion

The table II shows that The Exponential imperfect debugging model as illustrate for 3 Data Sets indicate that the model is performing well in arriving at a decision. Out of 3 Data Sets, the procedure applied on the model has given a decision of Accept for 2 and Continue for DS2 and no Rejection for the above Dataset at various times instant of the data as follows. Therefore, we may conclude that, applying SPRT on data sets we can come to an early conclusion of predicting a reliable / unreliable of software.

Acknowledgement

Our thanks to Department of Computer Science and Engineering and Department of Statistics of ,Acharya Nagarjuna University; for providing necessary facilities to carry out the research work.

References

[1] Card D., (1994), Statistical Process Control for Software, *IEEE Software*, May, 95-97.
[2] STIEBER, H.A.(1997). "Statistical Quality Control: How To Detect Unreliable Software Components", Proceedings the 8th International Symposium on Software Reliability Engineering, 8-12.
[3] John D.Musa; "Software Quality and Reliability Basics"; AT&T Bell Laboratories. CH 2468-7/87/0000/014,1987 IEEE.
[4] Wald. A., "Sequential Analysis", John Wiley and Son, Inc, New York. 1947.
[5] Dr. R Satya Prasad ,K Ramchand H Rao and Dr. R.R. L Kantham (2011)," Software Reliability Measuring using Modified Maximum Likelihood Estimation and SPC" IJCA Journal, Number 7 - Article 1.
[6] F. N. Chowdhiiry, John P. Christensen, Jorge L.Aravena, "Power System Fault detection and State Estimation Using Kalman Filt,er with Hypothesis Testing", IEEE Trans. on Power delivery, Vol. 6, No. 3, Jiily 1991,
[7] 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.
[8] Pham. H., "System software reliability", Springer. 2006.
[9] Xie, M., Goh. T.N., Ranjan.P., "Some effective control chart procedures for reliability monitoring" -Reliability engineering and System Safety 77 143 -150, 2002.
[10] Michael. R. Lyu, "The hand book of software reliability engineering", McGrawHill & IEEE Computer Society press
[11] 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.
[12] R. Satya Prasad and D. Haritha (2011). "Discovery of Reliable Software using GOM on Interval Domain Data" , International Journal of Computer Applications Volume 32– No. 5, pp. 7-12.