Research Paper
Impact Factor: 3.567
Peer Reviewed Journal

IJMDRR
E- ISSN –2395-1885
ISSN -2395-1877

BURR TYPE XII SOFTWARE RELIABILITY BY USING SPRT

Dr. B. Rama Devi
Asst. Professor in CSE, Vignan's Lara Institute of Technology and Science, Guntur.

Abstract
Almost everyone in the world is directly or indirectly affected by computer systems. Computers are embedded in diverse areas for various applications. Since the technology demand high- performance hardware and high quality software for making improvements. Software should not wear out and continue to operate after a bad result. We need reliable software for developing and testing products. Software reliability is defined as the probability of failure free software operation for a specified period of time in a specified environment (Musa, 1998). Software reliability assessment is increasingly important in developing and testing new software products. The reliability of software can be improved by understanding the characteristics of software and its design. Complete testing of the software is not possible; however sufficient testing & proper maintenance will improve software reliability to great extent.

Key words: Software Reliability, Burr Type XII, SPRT.

Introduction

Software Reliability is an important factor affecting system reliability. It differs from hardware reliability in that it reflects the design perfection, rather than manufacturing perfection. The high complexity of software is the major contributing factor of Software Reliability problems. Software Reliability is not a function of time - although researchers have come up with models relating the two. The modeling technique for Software Reliability is reaching its prosperity, but before using the technique, we must carefully select the appropriate model that can best suit our case. Measurement in software is still in its infancy. No good quantitative methods have been developed to represent Software Reliability without excessive limitations. Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability. Software Reliability is an important to attribute of software quality, together with functionality, usability, performance, serviceability, capability, install ability, maintainability, and documentation. Software Reliability is hard to achieve, because the complexity of software tends to be high. While any system with a high degree of complexity, including software, will be hard to reach a certain level of reliability, system developers tend to push complexity into the software layer, with the rapid growth of system size and ease of doing so by upgrading the software.

Many software models contain

1. Assumptions
2. Factors
3. Mathematical function

Software reliability can be divided into two categories

1. Prediction modeling and
2. Estimation modeling.

These modeling techniques follow observation and analyzes with statistical inference.

Prediction Model: This model uses historical data. They analyze previous data and some observations. They usually made prior development and regular test phases. The model follows the concept phase and the predication from the future time.

Estimation Model: Estimation model uses the current data from the current software development effort and doesn't use the conceptual development phases and can estimate at any time. There are many different proposals and it is hard to decide beforehand which one is the best fit. In addition, time measurement is a problem because software does not fail just because clock time passes, but it has to be used.

Objectives of the Study

1. To study the quality and reliability of software product by using SPRT.
2. To offer suggestions for effective use of Burr Type XII software.

Reliability Model

Reliability theory and models for estimating and predicting reliability have been developed for several decades. Reliability models are a very different type of quality models build with the Burr approach. They have a certain usage in practice and their application areas. We explain some basics, the general idea behind most reliability models is that we want to predict the future failure behavior – and thereby the reliability – of software based on actual failure data. That means that we use data

from failures as sample data in a stochastic model to estimate the model parameters. The failure data that comprises the sample data can be one of two types: (1) time between failure (TBF) data or (2) grouped data. The first type contains each failure and the time that has passed since the last failure. The second type has the length of a test or operation interval and the number of failures that occurred in that interval. The latter type is not as accurate as the first, but the data is easier to collect in real world projects.

A process which develops in time in accordance with some probabilistic laws is called a stochastic process. They are used for the description of a systems operation over time. The two main types of stochastic processes are continuous and discrete. Counting processes in reliability engineering are widely used to describe the appearance of events in time, e.g., failures, number of perfect repairs, etc. The simplest counting process is a Poisson process. Poisson-type models assume that the number of failures detected within distinct time intervals is independent.

Mathematically, reliability $R(t)$ is the probability that a system will be successful in the interval from time '0' to time 't':

$$R(t) = P(T > t) \qquad t \geq 0$$

Where, '*T* ' is a random variable which denotes the time-to-failure. Software reliability models serve to aid the software engineer by indicating the likelihood of system operation over a given time interval according to the stated specifications. Software reliability predictions can be used to judge the quality of a program. It is important to define the key terms in developing and describing software reliability models. The following definitions are used to distinguish between failures and faults throughout the thesis (Musa *et al*., 1987).

A **failure** is a departure from how software should behave during operation according to the requirements.
A **fault** is a defect in a program, that when executed causes failure(s).

Both static and dynamic software reliability models exist to assess the quality aspect of software. A static model uses software metrics, like complexity metrics, results of inspections, etc., to estimate the number of defects or faults in the software. Dynamic models use the past failure discovery rate during software execution or cumulative failure profile to estimate the number of failures. Because of this, they include a time component, which is typically based on recording times 't_i' of successive failure 'i' (i >0). Time may be recorded as execution time or calendar time. There is considerable statistical literature on modeling the reliability growth process of finding and fixing defects in a software product. Most software models contain assumptions, factors, and a mathematical function that relate the reliability with the factors. The mathematical function is usually exponential or logarithmic. Models differ based on their assumptions of the software and its execution environment. Some models use an NHPP model to the failure process. SRGMs are mathematical models that represent software failures as a random process and can be used to evaluate development status during testing. Many SRGMs have been proposed in the past four decades (Xie, 1991) for the estimation of reliability growth of products during software development processes. In general, SRGMs are applicable to the late stages of testing in software development.

Among these four categories of models the focus is on those which deal with the inter failure time (i.e., Time between failure models) and the random number of software failures in a given period of time (i.e., failure count models) of a developed software. The focal theme of the paper is to study the quality and reliability of a software product using SPRT.

Software reliability can be estimated once the mean value function is determined. The technique of control chart has been used in the software engineering so as to improve the quality of software products. Maximum Likelihood Estimation (MLE) and Sequential Probability Ratio Test (SPRT) which are extensively used in carrying out in this study.

Maximum Likelihood Estimation Method

The method of MLE is one of the most useful techniques for deriving point estimators. The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability of the sample data. A MLE method is versatile and applies to many models and to different types of data. Although the methodology for maximum likelihood estimation is simple, the implementation is mathematically intense.

If we conduct an experiment and obtain N independent observations, $T = t_1, t_2, \ldots, t_N$. The probability density function of 'k' unknown parameters $\theta_1, \theta_2, \ldots, \theta_k$ is given as $f(t; \theta_1, \theta_2, \ldots, \theta_k)$. Assuming that the random variables are independent, then the likelihood function, $L(T; \theta_1, \theta_2, \ldots, \theta_k)$, is the product of the probability density function evaluated at each sample point:

$$L(t_1, t_2, \ldots, t_N \mid \theta_1, \theta_2, \ldots, \theta_k) = L = \prod_{i=1}^{N} f(t_i; \theta_1, \theta_2, \ldots, \theta_k)$$

The maximum likelihood estimator $\hat{}$ is found by maximizing $L(T; \theta_1, \theta_2, \ldots, \theta_k)$ with respect to θ .
The symbol (^) is used here to distinguish Maximum Likelihood estimators from the parameters being used.
Likelihood function by using $\lambda(t)$ is expressed as, $L = \prod_{i=1}^{n} \lambda t_i$
Taking the natural logarithm of the above equation, we can obtain *lnL*.
The logarithmic likelihood function is given by, $Log\ L = log(\prod_{i=1}^{n} \lambda(t_i))$
The ML estimates (Cohen, 1965) of the unknown parameters $\theta_1, \theta_2, \ldots, \theta_k$ are obtained by maximizing and differentiating *lnL* with respect to each of the unknown parameters. Solving the equations simultaneously and equating to zero.

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

The log likelihood function for Interval domain data is given in Equation

Interval Domain Data

Assuming that the data are given for the cumulative number of detected errors y_i 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 LLF takes on the following form (Pham, 2006):

$$LLF = \sum_{i=1}^{n} (y_i - y_{i-1})\, log[m(t_i) - m(t_{i-1})] - m(t_n) \qquad (1.1)$$

Let S_k be the time between $(k-1)^{th}$ and k^{th} failure of the software product. Let X_k be the time up to the k^{th}failure. Let us find out the probability that time between $(k-1)^{th}$ and k^{th} failures, i.e., S_k exceeds a real number 's' given that the total time up to the $(k-1)^{th}$ failure is equal to x.

i.e., $P\left[S_k > \frac{s}{X_{k-1}} = x\right]$

$$R\, S_k/X_{k-1}(s/x) = e^{-[m(x+s)-m(s)]} \qquad (1.2)$$

This Expression is called Software Reliability.

Sequential Test for Software Reliability Growth Models

For the Poisson process we know that the expected value of $N(t) = \lambda(t)$ 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 such a process is

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

Depending on the forms of $m(t)$ we get various Poisson processes called NHPP, for our Burr type XII model. The mean value function is given as

$$m(t) = a\left[1 - \left(1 + t^c\right)^{-b}\right], \quad t \geq 0$$

We may write

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

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

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. The mean value function $m(t)$ contains the parameters $'a', 'b' and 'c'$. Let P_0, P_1 be values of the NHPP at two specifications of b say $b_0, b_1 \; where \; (b_0 < b_1)$ *and* two specifications of c say $c_0, c_1 \; where \; (c_0 < c_1)$. It can be shown that for our model $m(t)$ at b_1 is greater than that at b_0 and $m(t)$ *at* c_1 is greater than that at c_0. Symbolically $m_0(t) < m_1(t)$. Then the SPRT procedure is as follows:

Accept the system to be Reliable if $\frac{P_1}{P_0} \leq B$

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

$$i.e., \; N(t) \leq \frac{\log\left(\frac{\beta}{1-\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)} \qquad (1.3.1)$$

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

$$i.e., \quad \frac{e^{-m_1(t)}[m_1(t)]^{N(t)}}{e^{-m_0(t)}[m_0(t)]^{N(t)}} \geq A$$

$$i.e., \; N(t) \geq \frac{\log\left(\frac{1-\beta}{\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)} \qquad (1.3.2)$$

Continue the test procedure as long as

$$\frac{\log\left(\frac{\beta}{1-\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)} < N(t) < \frac{\log\left(\frac{1-\beta}{\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)} \qquad (1.3.3)$$

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

Acceptance Region

$$N(t) \leq \frac{\log\left(\frac{\quad}{(1-\sim)}\right) + a\left[\left(1+t^{c_0}\right)^{-b_0} - \left(1+t^{c_1}\right)^{-b_1}\right]}{\log a\left[\frac{\left(1+t^{c_0}\right)^{-b_0}}{\left(1+t^{c_1}\right)^{-b_1}}\right]} \qquad (1.3.4)$$

Rejection Region

$$N(t) \geq \frac{\log\left(\frac{1-}{\sim}\right) + a\left[\left(1+t^{c_0}\right)^{-b_0} - \left(1+t^{c_1}\right)^{-b_1}\right]}{\log a\left[\frac{\left(1+t^{c_0}\right)^{-b_0}}{\left(1+t^{c_1}\right)^{-b_1}}\right]} \qquad (1.3.5)$$

Continuation Region

$$\frac{\log\left(\frac{}{(1-\sim)}\right)+a\left[\left(1+t^{c_0}\right)^{-b_0}-\left(1+t^{c_1}\right)^{-b_1}\right]}{\log a\left[\frac{\left(1+t^{c_0}\right)^{-b_0}}{\left(1+t^{c_1}\right)^{-b_1}}\right]} < N(t) < \frac{\log\left(\frac{1-}{\sim}\right)+a\left[\left(1+t^{c_0}\right)^{-b_0}-\left(1+t^{c_1}\right)^{-b_1}\right]}{\log a\left[\frac{\left(1+t^{c_0}\right)^{-b_0}}{\left(1+t^{c_1}\right)^{-b_1}}\right]}$$

It may be noted that in the proposed 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, $m(t) = \lambda t$ the decision rules become decision lines as described by Stieber (1997). In that sense Equations (1.3.1), (1.3.2) , (1.3.3) 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.

SPRT Analysis of Datasets

In this section, the developed SPRT methodology is shown for a software failure data which is of interval domain. We evaluate the decision rules based on the considered mean value function for Six different data sets borrowed from (Pham 2005), (Wood 1996) and five different data sets, borrowed from Pham (2006), Zhang *et al*. (2002), Ohba (1984a), Misra (1983) are evaluated. Based on the estimates of the parameter 'b' in each mean value function, we have chosen the specifications of b_0 =b-¢ , b_1 =b+¢ and $c_0 = c - \delta, c_1 = c + \delta$ equidistant on either side of estimate of b obtained through a data set to apply SPRT such that $b0 < b < b1$ and $c_0 < c < c_1$. Assuming the value of ¢ = 0.5, the choices are given in the Table 1.4.1.

Table 1.1, Estimates of a, b, c & specifications of *b0, b1, c0, c1*

Dataset	*Estimate of 'a'*	*Estimate of 'b'*	b_0	b_1	*Estimate of 'c'*	c_0	c_1
Pham (2005) Phase 1Data	25.994042	0.978993	0.478993	1.478993	1.083116	0.583116	1.583116
Pham (2005) Phase 2 Data	41.590454	0.978993	0.478993	1.478993	1.083119	0.583119	1.583119
Wood (1996) Release #1 Data	87.533224	0.978352	0.478352	1.478352	1.082376	0.582376	1.582376
Wood (1996) Release #2 Data	111.778470	0.977674	0.477674	1.477674	1.081290	0.581290	1.581290
Wood (1996) Release #3 Data	59.376054	0.971698	0.471698	1.471698	1.051525	0.551525	1.551525
Wood (1996) Release #4 Data	42.831021	0.977671	0.477674	1.477674	1.081287	0.581287	1.581287
Dataset #1a	46.789258	0.978993	0.478993	1.478993	1.083120	0.583120	1.583120
Dataset #2a	25.994042	0.978993	0.478993	1.478993	1.083116	0.583116	1.583116
Dataset #3a	41.590454	0.978993	0.478993	1.478993	1.083119	0.583119	1.583119
Dataset #4a	74.532419	0.986064	0.486064	1.486064	1.077536	0.577536	1.577536
Dataset #5a	141.917892	0.986064	0.486064	1.486064	1.077536	0.577536	1.577536

Using the selected b_0, b_1 *and* c_0, c_1 and subsequently the $m_0(t)$ *and* $m_1(t)$ for each model we calculated the decision rules given by Equations (1.3.4), (1.3.5) sequentially at each 't' of the data set taking the strength (α, β) as (0.05, 0.2).These are presented for the model in Table 1.2.

Table 1.2, SPRT Analysis for 11 Datasets

Dataset	*T*	*N(t)*	*R.H.S. of Equation 5. 3.4 Acceptance Region (<=)*	*R.H.S. of Equation 5.3.5 Rejection Region (>=)*	*Decision*
Pham (2005) Phase 1Data	1	1	1.192028	1.856685	Accepted
Pham (2005) Phase 2 Data	1	3	1.792173	2.373032	Rejected
Wood (1996)Release #1 Data	1	16	3.338247	3.822450	Rejected
Wood (1996) Release #2 Data	1	13	4.089652	4.588755	Rejected
Wood (1996) Release #3 Data	1	6	2.430323	2.960544	Rejected
Wood (1996) Release #4 Data	1	1	1.839248	2.415563	Accepted
Dataset #1a	1	2	1.979767	2.542835	Rejected
	2	3	1.173742	1.398759	
Dataset #2a	1	1	1.192028	1.856685	Accepted
Dataset #3a	1	3	1.792173	2.373032	Rejected
Dataset #4a	1	6	2.905064	3.407325	Rejected
Dataset #5a	1	9	4.954790	5.391775	Rejected

It is observed from the Table1.2 that a decision either to accept or reject the system is reached much in advance of the last time instant of the data.

Conclusion

The Table 1.2 of Interval domain data as exemplified for 11 Data Sets shows that Burr Type XII model is performing well in arriving at a decision. Out of 11 Datasets the procedure applied on the model has given a decision of rejection for 8 datasets, acceptance for 3 datasets and continue for none at various time instant of the data. The datasets Phase1, Release #4 and Dataset #2a are accepted at 1st instance of time whereas remaining datasets are rejected at different instances of time. Therefore, by applying SPRT on data sets it can be concluded that we can come to an early conclusion of reliable or unreliable.

References

1. Lyu, M.R. and Nikora, A.(1991), "*Heuristic Approach for Software Reliability Prediction: The Equally Weighted Linear Combination Model"*, Proceedings of the International Symposium on Software Reliability Engineering, Texas, 172-181.
2. Lyu, M.R.(1996), "*The Hand Book of Software Reliability Engineering*", McGrawHill & IEEE Computer Society Press.
3. Macgregor, J.F. and Koutri, T., (1995). "*Statistical process control of multivariate processes*", Control Eng. Practice, Vol. 3, No. 3, pp 403-414.
4. Malaiya, Y. K., Sumit sur, Karunanithi, N. and Sun, Y. (1990). "Implementation considerations for software reliability", 8th Annual Software Reliability Symposium, June 1.
5. Misra, P. (1983). "Software Reliability Analysis," IBM Systems Journal, vol. 22, pp. 262–270.
6. Montgomery, D. C. and Woodall, H. L. (1997). "*A Discussion on Statistically-Based Process Monitoring and Control*", Journal of Quality Technology, 29(2), 121-162.
7. Musa, J.D. (1971), "*A Theory of Software Reliability and its Application*", IEEE Trans. Software Eng., Vol. SE-1, 312-327.
8. *Dr. R. Satya Prasad, B.Ramadevi and Dr. G. Sridevi* "Detection of Burr type XII Reliable Software Using SPRT on Interval Domain Data*", International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 6 (2), 2015, 1806-1811*
9. Sridevi G and Rama Devi B, Software Quality Assessment: Using Burr Type XII Distribution Model – August 8, 2016, ISBN-13: 978-3659840524-Book.