European Journal of Applied Sciences 8 (1): 01-09, 2016
ISSN 2079-2077
© IDOSI Publications, 2016
DOI: 10.5829/idosi.ejas.2016.8.1.22841

Model Comparison with SPC

Gutta Sridevi

Department of CSE, KL University, Guntur, AP., India

Abstract: One of the important quality metric for any developed software is its reliability. The aim of the software reliability is reducing or eliminating failures in software systems. This is estimated by the use of any statistical model whose unknown parameters are estimated from the available software failure data. A few researchers develop SPC based software reliability monitoring techniques to improve software reliability process. This paper presents a model comparison with control chart mechanism based on interval domain data using the mean value functions of Pareto type IV and Burr type XII distributions, which are then based on Non Homogenous Poisson Process (NHPP).

Key words: NHPP · Pareto type IV model · Burr type XII model · ML estimation · SPC

INTRODUCTION

Today software reliability assessment is important to evaluate, since it is the main attribute of software. During the past three decades research on software reliability engineering has been conducted and developed numerous statistical models for estimating software reliability. Most existing models for predicting software reliability are based purely on the observation of software product failures where they require a considerable amount of failure data to obtain an accurate reliability prediction. The reliability of the software is usually estimated by deriving mathematical models, describing a typical behavior of a debugging process. The future failure behavior of a software system is predicted by studying and modeling its past failure behavior [1-3].

To assess the reliability of software, many researchers have been conducted research activities and a number of NHPP software reliability growth models have been proposed. The technique of control chart has been used in the software engineering so as to improve the quality of software products. A work applying statistical quality control in measuring software reliability is carried by [4].

SRGMs can estimate the number of initial faults, the software reliability, the failure intensity, the mean time-interval between failures, etc. SRGMs are useful for estimating how software reliability improves as faults are detected and repaired. It can be used to predict when a particular level of reliability is likely to be attained and also helps in determining when to stop testing to attain a given reliability level. SRGMs help in decision making in many software development activities such as number of initial faults, failure intensity, reliability within a specified interval of time period, number of remaining faults, cost analysis and release time etc.

Statistical Process Control (SPC) is an analytical decision making tool for monitoring and controlling manufacturing processes. SPC determines when a statistically significant change has taken place in the process or when a seemingly significant change is just due to chance causes. SPC involves:

- Determining the critical process parameters that need to be monitored
- Setting up an initial control chart and confirming that the process is in-control and
- Collecting and plotting future data on the chart and interpreting the chart to determine if the process has gone out-of-control.

To improve the manufacturing process and to reduce the variation, control charts and other SPC techniques are used in the companies. Sometimes companies implement SPC to satisfy customer requirements or to meet certification requirements. SPC is a type of charting that tells about the variation that exists in the systems. SPC tools can be better adapted to a software product also with necessary amendments [5].

Corresponding Author: Gutta Sridevi, Department of CSE, KL University, Guntur, AP., India.

S - Statistical, because we use some statistical concepts to help us understand processes.
P - Process, because we deliver our work through processes. i.e. how we do things.
C - Control by this we mean predictable.

The application of SPC involves three main phases of activity:

- Understanding the process and the specification limits.
- Eliminating assignable (special) sources of variation, so that the process is stable.
- Monitoring the on-going production process, assisted by the use of control charts, to detect significant changes of mean or variation.

The Utilization of SPC for software reliability has been the subject of study for several researchers. The studies are based on reliability process improvement models. SPC is used as a means of accomplishing high process maturities. Some of the studies provide guidelines in using SPC by modifying general SPC Principles so that it can be suitable for the special requirements of software development [6, 7]. It is especially noteworthy that Burr and Owen provided seminal guidelines by outlining the techniques that can be used for managing and controlling the reliability of software. The use of control charts is significant and is considered to be one of the tools for SPC.

Background: Burr type XII distribution was first introduced in 1942 by Irving W. Burr [8]. Since the corresponding density functions have a wide variety of shapes, this system is useful for approximating histograms. The Burr XII (BXII) distribution is a very popular distribution for modeling lifetime data and for modeling phenomenon with monotone failure rates. It has been applied in the field of reliability studies and failure time modeling. Burr type XII distributions were further investigated by [8, 9]. If 't' is a continuous random variable with pdf: $f(t;\ \theta_1,\theta_2,...,\theta_k)$. Where $\theta_1,\theta_2,...,\theta_k$, are k unknown constant parameters which need to be estimated and CDF: $F(t)$ Where, the mathematical relationship between the PDF and CDF is given by: $f(t)=\frac{d(F(t))}{dt}$. Let 'a' denote the number of expected faults that would be detected given infinite testing time in case of finite failure NHPP models. Then, the mean value function of the finite failure NHPP models can be written as: $m(t) = aF(t)$. Where, $F(t)$ is a cumulative distributive function. The failure intensity function $\lambda(t)$ in case of the finite failure NHPP models is given by: $\lambda(t) = aF(t)$ [10].

NHPP Model Assumptions: There are several software reliability growth models available for use according to probabilistic assumptions. The first one is the Markovian model which is the failure process represented by Markov. The second one is the fault counting model which describes the failure phenomenon by stochastic process like Homogeneous Poisson Process (HPP), Non Homogeneous Poisson Process (NHPP) and Compound Poisson Process. The Non Homogenous Poisson Process (NHPP) based software reliability growth models are proved to be quite successful in practical software reliability engineering [11]. Model parameters can be estimated by using maximum Likelihood Estimation (MLE). The formulation of NHPP model is described in the following lines.

A software system is subject to failures at random times caused by errors present in the system. Let $\{N(t), t \geq 0\}$ be the cumulative number of software failures by time 't', where t is the failure intensity function, which is proportional to the residual fault content. As there will be no errors at t=0 we have

$$N(t) = 0$$

Let $m(t)$ represent the expected number of software failures by time't'. As the expected number of errors remaining in the system is finite, the mean value function $m(t)$ is finite.

$$m(t) = \begin{cases} 0, t = 0 \\ a, t \to \infty \end{cases}$$

where 'a' is the expected number of software errors to be eventually detected. Suppose $N(t)$ is known to have a Poisson probability mass function with parameters $m(t)$ i.e.,

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

where N(t) is the cumulative number of failures observed by time 't', N(t) can be modeled as a Poisson Process with a time dependent failure rate. Thus the stochastic behavior of software failure phenomena can be described through the N(t) process.

Proposed Models: This section presents two types of distribution models for comparison with SPC - Burr type XII distribution model [12] and a Pareto type IV model [13].

Burr type XII Model Development: The Cumulative distributive function (CDF) for Burr type XII is given by [14].

$$m(t) = \int_0^1 \lambda(t)\,dt = a\left[1-\left(1+t^c\right)^{-b}\right]$$
$$= a\,F(t)$$

The Probability Density Function (PDF) of Burr XII distribution are given, respectively by

$$\lambda(t) = a\left[\frac{cbt^{c-1}}{\left(1+t^c\right)^{b+1}}\right] = a\,f(t)$$

Mathematical Derivation for Parameter Estimation – Burr type XII Model: The Log Likelihood function of Interval domain data is given by:

$$LogL = \sum_{i=1}^{k}\left(n_i - n_{i-1}\right)\log\left[m(t_i) - m(t_{i-1})\right] - m(t_k) \tag{1}$$

Take the mean value function of Burr Type XII is of the form

$$m(t) = a\left[1-(1+t^c)^{-b}\right] \tag{2}$$

By substituting Equation (2) in the above Equation (1), we get

$$LogL = \sum_{i=1}^{k} \begin{matrix}\left(n_i - n_{i-1}\right)\log\left\{a\left[1-\left(1+t_i^c\right)^{-b}\right] - a\left[1-\left(1+t_{i-1}^c\right)^{-b}\right]\right\} \\ -a\left[1-\left(1+t_k^c\right)^{-b}\right]\end{matrix}$$

$$= \sum_{i=1}^{k}\left(n_i - n_{i-1}\right)\log\text{ a}\left[\left(1+t_{i-1}^c\right)^{-b} - \left(1+t_i^c\right)^{-b}\right]$$

$$-a + a\left(1+t_k^c\right)^{-b}$$

$$Log\,L = \sum_{i=1}^{k} \begin{matrix}\left(n_i - n_{i-1}\right)\left\{Log\,a + Log\left[\left(1+t_{i-1}^c\right)^{-b} - \left(1+t_i^c\right)^{-b}\right]\right\} \\ -a + a\left(1+t_k^c\right)^{-b}\end{matrix} \tag{3}$$

The parameter 'a' is estimated by taking the partial derivative of Log Lw.r.t 'a' and equating to '0'. $\left(i.e., \frac{\partial\, Log\, L}{\partial a} = 0\right)$

$$\frac{\partial LogL}{\partial a} = \sum_{i=1}^{k}\left(n_i - n_{i-1}\right)\left[\frac{1}{a} + 0\right] - 1 + \left(1+t_k^c\right)^{-b}$$

$$\therefore a = \sum_{i=1}^{k}\left(n_i - n_{i-1}\right)\frac{\left(1+t_k^c\right)^b}{\left(1+t_k^c\right)^b - 1} \tag{4}$$

The parameter 'b' is estimated by iterative Newton Raphson Method using $b_{n+1} = b_n - \frac{g(b)}{g'(b)}$, Where $g(c)$ *and* $g'(c)$ are expressed as follows.

$g(b) = \frac{\partial LogL}{\partial b} = 0$

$$\frac{\partial Log\,L}{\partial b} = g(b) = \sum_{i=1}^{k} (\mathrm{n}_i - n_{i-1}) \left\{ \begin{array}{l} \left[-\log\left(t_{i-1}+1\right) - \log(t_i+1) + \dfrac{(t_i+1)^b \log(t_i+1) - (t_{i-1}+1)^b \log(t_{i-1}+1)}{(t_i+1)^b - (t_{i-1}+1)^b} \right] \\ + \left[\dfrac{1}{(t_k+1)^b - 1} Log\left(\dfrac{1}{1+t_k} \right) \right] \end{array} \right\} \tag{5}$$

Again partial differentiating with respect to 'b' and equate to 0, we get

$$g^{'}(b) = \frac{\partial^2 LogL}{\partial b^2} = 0$$

$$\frac{\partial^2 LogL}{\partial b^2} = g^{'}(b) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[\frac{2(t_{i-1}+1)^b (t_i+1)^b \log(t_i+1) \log\left(\dfrac{t_{i-1}+1}{t_i+1} \right)}{\left[(t_i+1)^b - (t_{i-1}+1)^b \right]^2} \right] + \sum_{i=1}^{k} (n_i - n_{i-1}) \log(1+t_k) \frac{(t_k+1)^b \log(t_k+1)}{\left[(t_k+1)^b - 1 \right]^2} \tag{6}$$

The parameter 'c' is estimated by iterative Newton Raphson Method using
$c_{n+1} = c_n - \frac{g(c_n)}{g'(c_n)}$ Where $g(c)$ *and* $g'(c)$ are expressed as follows.

$$g(c) = \frac{\partial LogL}{\partial c} = 0$$

$$\frac{\partial LogL}{\partial c} = g(c) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[-\log t_{i-1} \frac{t_{i-1}^c}{(1+t_{i-1}^c)} - \log t_i \frac{t_i^c}{(1+t_i^c)} + \frac{t_i^c \log t_i - t_{i-1}^c \log t_{i-1}}{(t_i^c - t_{i-1}^c)} \right] - \sum_{i=1}^{k} (\mathrm{n}_i - n_{i-1}) \frac{\log t_k}{(1+t_k^c)} \tag{7}$$

$$g'(c) = \frac{\partial^2 LogL}{\partial c^2} = 0$$

$$\frac{\partial^2 Log\,L}{\partial c^2} = g^{'}(c) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[\begin{array}{l} \left(Log\left(\dfrac{t_{i-1}}{t_i} \right) \dfrac{t_i^c . t_{i-1}^c}{(t_i^c - t_{i-1}^c)^2} \left\{ \log t_i - \log t_{i-1} \right\} \right) \\ -(\log t_{i-1})^2 . \dfrac{t_{i-1}^c}{(1+t_{i-1}^c)^2} - (\log t_i)^2 \dfrac{t_i^c}{(1+t_{i-1}^c)^2} \end{array} \right] + \sum_{i=1}^{k} (n_i - n_{i-1})(\log t_k)^2 . \frac{t_k^c}{(1+t_k^c)^2} \tag{8}$$

Pareto type IV Model Development: R.Satya Prasad (2007) studied some problems of software reliability prediction and analysis when the random phenomenon in a software failure data with a half logistic distribution as means value functions. Kantam R.R.L and Subbarao (2009) developed a SRGM using Pareto (IV) distribution[15-20].

The Log Likelihood function is given as:

$$Log\ L = \sum_{i=1}^{k} (n_i - n_{i-1}). log[m(t_i) - m(t_{i-1})] - m(t_k) \tag{9}$$

Take the mean value function of Pareto Type IV is of the form

$$m(t) = a\left[1 - \left[1 + \left(\frac{t}{c}\right)\right]^{-b}\right] \tag{10}$$

By substituting Equation (10) in Equation (9), we get

$$Log\ L = \sum_{i=1}^{k}(n_i - n_{i-1})\log\left[a\left(1-\left\{1+\left(\frac{t_i}{c}\right)\right\}^{-b}\right) - a\left(1-\left\{1+\left(\frac{t_{i-1}}{c}\right)\right\}^{-b}\right)\right] - a\left[1-\left\{1+\left(\frac{t_k}{c}\right)\right\}^{-b}\right]$$

$$Log\ L = \sum_{i=1}^{k}(n_i - n_{i-1})\log\left[\left(a - a\left\{1+\left(\frac{t_i}{c}\right)\right\}^{-b}\right) - a + a\left\{1+\left(\frac{t_{i-1}}{c}\right)\right\}^{-b}\right] - a + a\left\{1+\left(\frac{t_k}{c}\right)\right\}^{-b}$$

$$= \sum_{i=1}^{k}(n_i - n_{i-1})\log\left[a\left\{1+\left(\frac{t_{i-1}}{c}\right)\right\}^{-b} - a\left\{1+\left(\frac{t_i}{c}\right)\right\}^{-b}\right] - a + a\left\{1+\left(\frac{t_k}{c}\right)\right\}^{-b}$$

$$Log\ L = \sum_{i=1}^{k}(n_i - n_{i-1})\left[Log\ a + Log\left(\left\{\frac{c+t_{i-1}}{c}\right\}^{-b} - \left\{\frac{c+t_i}{c}\right\}^{-b}\right)\right] - a + a\left\{\frac{c+t_k}{c}\right\}^{-b} \tag{11}$$

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

$$\frac{\partial Log\ L}{\partial a} = \sum_{i=1}^{k}(n_i - n_{i-1})\left[\frac{1}{a} + 0\right] - 1 + \left(\frac{c+t_k}{c}\right)^{-b}$$

$$\therefore\ a = \sum_{i=1}^{k}(n_i - n_{i-1})\frac{(c+t_k)^b}{(c+t_k)^b - c^b} \tag{12}$$

$$g(b) = \frac{\partial LogL}{\partial b} = \sum_{i=1}^{k}(n_i - n_{i-1})\left[-\log(t_{i-1}+1) - \log(t_i+1) + \left\{\frac{(t_i+1)^b\log(t_i+1) - (t_{i-1}+1)^b\log(t_{i-1}+1)}{(t_i+1)^b - (t_{i-1}+1)^b}\right\}\right]$$
$$+ \sum_{i=1}^{k}(n_i - n_{i-1})\frac{1}{(t_k+1)^b - 1}\log\left(\frac{1}{t_k+1}\right) \tag{13}$$

$$g^{'}(b) = \frac{\partial^2 Log\,L}{\partial b^2} = \sum_{i=1}^{k}(n_i - n_{i-1})\left[\frac{2(t_{i-1}+1)^b(t_i+1)^b\,Log(t_i+1)Log\left(\frac{t_{i-1}+1}{t_i+1}\right)}{\left[(t_i+1)^b - (t_{i-1}+1)^b\right]^2}\right] +$$
$$\left[\sum_{i=1}^{k}(n_i - n_{i-1})Log(t_k+1)\left\{\frac{(t_k+1)^b\,Log(t_k+1)}{\left[(t_k+1)^b - 1\right]^2}\right\}\right] \tag{14}$$

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

$$c_{n+1} = c_n - \frac{g(c_n)}{g'(c_n)}$$

Table 1: Dataset Release #1 from Alan Wood Tandem Computers -1996

Test Week	CPU Hours	Percent CPU Hours	Defects Found	Predicted Total Defects
1	519	-	16	-
2	968	-	24	-
3	1,430	-	27	-
4	1,893	-	33	-
5	2,490	-	41	-
6	3,058	-	49	-
7	3,625	-	54	-
8	4,422	-	58	-
9	5,218	-	69	-
10	5,823	58	75	98
11	6,539	65	81	107
12	7,083	71	86	116
13	7,487	75	90	123
14	7,846	78	93	129
15	8,205	82	96	129
16	8,564	86	98	134
17	8,923	89	99	139
18	9,282	93	100	138
19	9,641	96	100	135
20	10,000	100	100	133

Where $g(c)$ and $g'(c)$ are expressed as follows.

$$g(c) = \frac{\partial LogL}{\partial c} = 0$$

$$g(c) = \frac{\partial LogL}{\partial c} = \sum_{i=1}^{k}(n_i - n_{i-1})\left[\frac{1}{c} - \frac{1}{(t_{i-1}+c)} - \frac{1}{(t_i+c)}\right] + \sum_{i=1}^{k}(n_i - n_{i-1})\left\{\frac{1}{(t_k+c)}\right\} \tag{15}$$

Taking the partial derivative again w.r.t ‘c’ and equating to ‘0’.

$$g^{'}(c) = \frac{\partial^2 LogL}{\partial c^2} = 0$$

$$\therefore g^{'}(c) = \sum_{i=1}^{k}(n_i - n_{i-1})\left[\frac{1}{(t_{i-1}+c)^2} + \frac{1}{(t_i+c)^2} - \frac{1}{c^2}\right] - \sum_{i=1}^{k}(n_i - n_{i-1})\left\{\frac{1}{(t_k+c)^2}\right\} \tag{16}$$

Data Analysis and Results: Based on the interval domain data given in Table 1[17], we compute the software failures process through Mean Value Control chart. We used cumulative interval domain data for software reliability monitoring using Burr type XII and Pareto type IV distributions. The parameters ‘a’, ‘b’ and ‘c’ are computed using iterative method for the given cumulative interval domain failures data shown in Table 1. Using ‘a’, ‘b and ‘c’ values we can compute $m(t)$.

Calculation of Control Limits for Burr type XII Model: The control limits for the chart are defined in such a manner that the process is considered to be out of control when the time to observe exactly one failure is less than LCL or greater than UCL. Our aim is to monitor the failure process and detect any change of the intensity parameter. When the process is normal, there is a chance for this to happen and it is commonly known as false alarm. The traditional false alarm probability is to set to be 0.27% although any other false alarm probability can be used. The actual acceptable false alarm probability should in fact depend on the actual product or process [18].

Table 2: Parameter estimates and Control limits of Interval domain data

		Estimated Parameters			Control Limits		
Model	Sample size	a	B	C	UCL	CL	LCL
Burr XII	20	87.533224	0.978352	1.082376	87.4150541	43.766612	0.1181698
Pareto IV	20	123.84453	0.978352	9.144224	123.677344	61.922267	0.167190

Table 3: Successive differences of mean values of Dataset Release #1 – Burr type XII

TT (day)	CF	m(t)	Successive Differences	TT (day)	CF	m(t)	Successive Differences
1	16	83.10273349	1.498292262	11	81	86.70612791	0.050426019
2	24	84.60102575	0.334379391	12	86	86.75655393	0.036226805
3	27	84.93540514	0.486249694	13	90	86.79278073	0.025083141
4	33	85.42165484	0.425949234	14	93	86.81786387	0.023482679
5	41	85.84760407	0.285704068	15	96	86.84134655	0.014839376
6	49	86.13330814	0.135073489	16	98	86.85618593	0.00718997
7	54	86.26838163	0.091055768	17	99	86.8633759	0.007043023
8	58	86.35943739	0.195173996	18	100	86.87041892	0
9	69	86.55461139	0.081936876	19	100	86.87041892	0
10	75	86.63654827	0.069579644	20	100	86.87041892	-----

Table 4: Successive differences of mean values of Dataset Release #1 – Pareto type IV

TT (day)	CF	m(t)	Successive Differences	TT (day)	CF	m(t)	Successive Differences
1	16	77.808782	10.902146	11	81	110.643761	0.679099
2	24	88.710928	2.855628	12	86	111.322861	0.494472
3	27	91.566557	4.503169	13	90	111.817333	0.345705
4	33	96.069727	4.343190	14	93	112.163039	0.326185
5	41	100.412917	3.159073	15	96	112.489225	0.207416
6	49	103.571990	1.571890	16	98	112.696641	0.100862
7	54	105.143881	1.090659	17	99	112.797504	0.099033
8	58	106.234540	2.429100	18	100	112.896537	0
9	69	108.663641	1.059895	19	100	112.896537	0
10	75	109.723536	0.920224	20	100	112.896537	------

$$T_U = \left[1-\left(1+t^c\right)^{-b}\right] = 0.99865$$

$$T_C = \left[1-\left(1+t^c\right)^{-b}\right] = 0.5$$

$$T_L = \left[1-\left(1+t^c\right)^{-b}\right] = 0.00135$$

Calculation of Control Limits for Pareto type IV Model:

$$T_U = \left[1-\left(1+\frac{t}{c}\right)^{-b}\right] = 0.99865$$

$$T_C = \left[1-\left(1+\frac{t}{c}\right)^{-b}\right] = 0.5$$

$$T_L = \left[1-\left(1+\frac{t}{c}\right)^{-b}\right] = 0.00135$$

The estimated parameters and the control limits are shown in Table 2.

Tables 3 and 4 shows the mean value function and their corresponding successive differences of the proposed two models respectively.

Fig. 1 and 2 are obtained by placing the interval domain failures cumulative data shown in Tables 3, 4 on y axis and failure number on x axis and the values of control limits are placed on Mean Value chart. The Mean Value chart of Burr type XII shows that the 7th and 9th failure data has fallen below $m(t_L)$. The Mean Value chart of Pareto type IV shows that the 16th and 17th failure data has fallen below $m(t_L)$. The successive differences of mean values below $m(t_L)$ indicates the failure process. In the present scenario, it is significantly stated that early detection of failure detection is done through Burr type XII model. The software quality is determined by detecting failures at an early stage. No failure data fall outside the $m(t_U)$. It does not indicate any alarm signal.

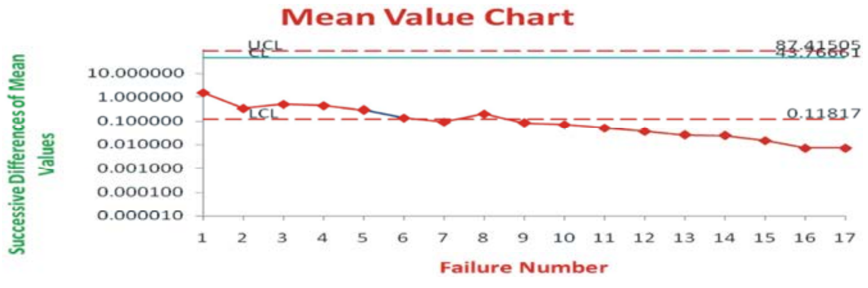

Fig. 1: Burr type XII Mean Value Chart for Dataset Release #1

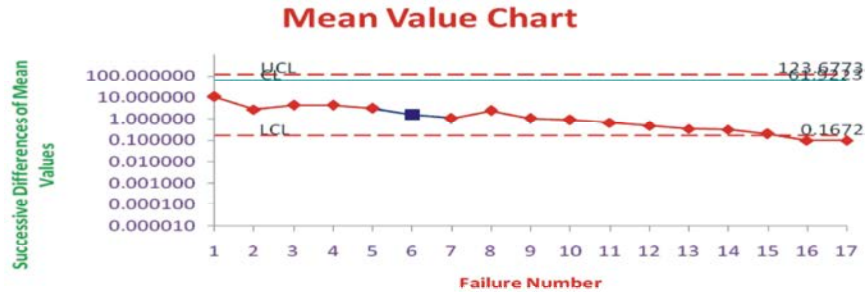

Fig. 2: Pareto type IV Mean Value Chart for Dataset Release #1

CONCLUSION

The given 20 interval domain failure data are plotted through the estimated mean value function against the failure serial order. The estimation of parameters is carried out by Newton Raphson Iterative method for the models. The graphs have shown out of control signals i.e below the $m(t_L)$ or LCL. Hence we conclude that our method of estimation and the control chart are giving a +ve recommendation for their use in finding out preferable control process or desirable out of control signal. We identified that by observing the Mean value Control chart the failure situation is detected at 7th and 9th point of Table 3, 16th and17th point of Table 4 i.e failure data has fallen below $m(t_L)$. The successive difference of mean values below $m(t_L)$ indicates the failure process. In the present scenario, it is proved that an early detection of failure through Burr type XII using the Mean Value Chart. The software quality is determined by detecting failures at an early stage for the corresponding $m(t)$. The early detection of software failure will improve the software reliability.

REFERENCES

1. Yamada, S., J. Hishitani and S. Osaki, 1993. Software reliability growth with a weibull test effort, IEEE Transactions on Reliability, 42:100-106.
2. Yamada, S. and S. Osaki, 1985. Software Reliability growth modeling: Models and Applications, IEEE Transactions on Software Engineering, 11:1431-1437.
3. Musa, J.D., 1998. Software Reliability Engineering, McGraw-Hill.
4. Stieber, H.A., 1997. Statistical Quality Control: How To Detect Unreliable Software Components, Proceedings the 8th International Symposium on Software Reliability Engineering, pp: 8-12.
5. John Oakland, 2008. Statistical Process Control, Sixth Edition, Butterworth-Heinemann, Elsevier.
6. Komil Umit Sargut, 2003. Application of SPC to Software Development Process via control charts, Thesis submitted to the Graduate School of Informatics, The Middle East Technical University.
7. Cartleton, Florac and A.William, 1999. Measuring the Software Process. Reading, MA:Addison-Wesley.
8. Burr, 1942.Cumulative Frequency Functions, Annals of Mathematical Statistics, 13: 215-232.
9. Hatke, Mary Agnes, 1949. A Certain Cumulative Probability Function. Ann. Math. Statist. 20(3) :461—463.
10. Pham, H., 2006. System Software Reliability, Springer.
11. Musa, J.D., A. Iannino and k. Okumoto, 1987. Software Reliability: Measurement Prediction Application. McGraw -Hill, New York.

12. Satya Prasad, R., B. Rama Devi and G. Sridevi, 2015. Assessing Burr Type XII Software Reliability for Interval Domain Data using SPC, Elixir Comp. Sci. & Engg.. ISSN: 2229-712X, 79 :30335-30340.
13. Sridevi G.and R.Satya Prasad, 2013. Monitoring Pareto Type IV SRGM Using SPC", International Journal of Computers and Technology (IJCT),, September, Published by Council for Innovative Research, ISSN, 11(1): 12277-3061.
14. Satya prasad, R., K.V. Murali Mohan and G. Sridevi, 2014. Burr type XII Software Reliability Growth Model", International Journal of Computer Applications, pp: 108
15. Satya Prasad, R.,2007. Half Logistic Software Reliability Growth Model, Ph.D Thesis of ANU, India.
16. Kantam, R.R.L. and R. Subbarao, 2009. Pareto Distribution: A Software Reliability Growth Model", International Journal of Performability Engineering,5(3): 275- 281.
17. Wood, A., 1996.Predicting Software Reliability, IEEE Computer, 2253-2264.
18. Xie, M., T.N.Goh. and P. Ranjan, 2002. Some effective control chart procedures for reliability monitoring-Reliability engineering and System Safety, 77: 143 - 150.
19. Pham. H., 2003. Handbook Of Reliability Engineering, Springer.
20. Lyu, M.R., 1996.Handbook of Software Reliability Engineering, McGraw-Hill, New York.