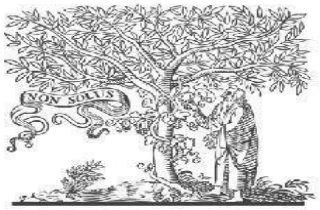
International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

COPY RIGHT



ELSEVIER SSRN

IJIEMR Transactions, online available on 21st Mar 2023. Link :http://www.ijiemr.org/downloads.php?vol=Volume-12&issue=Issue 03

10.48047/IJIEMR/V12/ISSUE 03/114

Title **SPC AND ORDER STATISTICS: PARETO TYPE IV MODEL**

Volume 12, ISSUE 03, Pages: 819-828

Paper Authors **B.N.V.Uma Shankar, K.Rosaiah**

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Guidelines We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

SPC AND ORDER STATISTICS: PARETO TYPE IV MODEL

B.N.V.Uma Shankar , Research Scholar and Senior Grade Asst.Prof, Department of Basic Sciences & Humanities, CVR College of Engineering, Hyderabad, India
K.Rosaiah, Department of Statistics, Acharya Nagarjuna University, India

ABSTRACT:

Over the years, numerous software defect prediction models have been developed to tackle the challenges inherent in software project development. Emphasizing software reliability is crucial for enhancing overall software quality, as it involves the analysis and projection of software quality based on defect prediction. Many software enterprises are actively striving to enhance software quality while concurrently reducing software development expenses. Among the diverse models available, the Pareto Type IV model stands out as a pivotal approach for scrutinizing software flaws using generated data. Complementing this, Statistical Process Control (SPC) emerges as a statistical technique offering contextual quality assurance. In this research endeavor, an enhanced software defect prediction model takes center stage, serving to anticipate errors that materialize during distinct phases of software development. This innovative model seeks to leverage insights from historical data, refined algorithms, and a comprehensive understanding of influencing factors. Through such proactive measures, the goal is to optimize software quality, minimize defects, and streamline the software development lifecycle.

KeyWords: Pareto type IV distribution model, NHPP, MLE method, Statistical Process Control, Order Statistics.

INTRODUCTION:

The concept of software reliability emerged as a response to the growing importance of software systems in various industries and the realization that software failures could lead to significant economic losses, safety hazards, and disruptions. As computers became more powerful and software systems grew in complexity, software began to be used in critical applications such as aerospace, defense, healthcare, and finance. Failures in these systems could have catastrophic consequences, highlighting the need for increased focus on reliability. Researchers began to develop mathematical models to analyze and predict software reliability, drawing inspiration from reliability engineering principles used in hardware systems. The concept of "software aging" gained traction during this decade. It was observed that software systems often became less reliable over time due to factors like resource leaks, memory corruption, and performance degradation. This led to the development of reliability growth models that aimed to track and improve software reliability over its life cycle. SRM, also referred to as the numerical model in this context, plays a crucial role in enhancing error detection subsequent to code modifications [1]. These models primarily concentrate on determining the necessary testing efforts designated for [2].

The growth of software-intensive systems, including embedded systems, web applications, mobile apps, and cloud

computing, further emphasized the importance of software reliability. Agile and DevOps methodologies introduced new challenges and opportunities for integrating reliability practices into the software development life cycle. Today, software reliability is an integral part of software engineering and quality assurance practices. Techniques like continuous integration, automated testing, and continuous monitoring help in identifying and addressing reliability issues throughout the development process. The frequency of errors that occur while debugging software is the basis for several software reliability models. In reliability engineering, the NHPP is particularly useful for modeling the failure rates of systems or components that change over time due to various factors such as environmental conditions, usage patterns, and design changes. The key idea behind NHPP is that the failure rate is not constant but varies as a function of time.

In software projects, failures are a common occurrence during both the development and execution phases [3]. Determining software reliability can be a challenging endeavor due to the inherent complexity of software types. Identifying an appropriate reliability model often hinges on the selection of reliable models. The notion of estimated software lacks a definitive description [4]. In cases where direct reliability measurement is not feasible, assessing reliability attributes provides an alternative avenue.

The NHPP model incorporates an "intensity function" or "failure rate function" that describes how the failure rate changes over time. This function can take various forms depending on the specific characteristics of the system being analyzed. The NHPP model offers the capability to estimate the anticipated quantity of failures occurring within a specific timeframe or to forecast the prospective failure rates of a system by drawing insights from historical data.

Order statistics are useful in a wide range of scenarios. Many books show how they are used in characterization problems, outlier identification, life-testing, data compression, linear estimation, system reliability studies, survival analysis, and many other domains. This study proposes a control mechanism derived from the mean value function of a Pareto type IV distribution, employing a Non-Homogeneous Poisson Process (NHPP) as its foundation. The control strategy is devised using order statistics applied to the cumulative quantity observed in time-based domain failure data. The research employs live datasets to validate the Pareto type IV distribution model in conjunction with the order statistics technique. The paper concludes by summarizing its findings and implications.

RELATED WORK

In a study by G. Sridevi et al., [5], they introduced the utilization of the Burr Type XII distribution model to assess software quality through Statistical Process Control (SPC). Meanwhile, B. Amulya and their coauthors [6] applied Statistical Process Control (SPC) for failure evaluation and proposed the use of the Pareto Type II distribution model along with an order statistic approach. The assessment of High-Level Design (HLD) was carried out using NHPP-based time-domain data. This approach combines both models to analyze failure data and employs control charts to visualize performance.

Furthermore, Y. Wu and their team [7] recommended an advanced software testing strategy that highlights the SRAT (Software Reliability Assessment Tool) as a more efficient method for generating reliable testing outcomes. This strategy

also assesses the functionality and programs specified within the software code.

In their study, M. Nafreen et al., [8] introduced a novel approach that combines software defect tracking with the Software Reliability Growth Model (SRGM). This integrated methodology was applied within a NASA project to identify and analyze a multitude of software flaws and their subsequent resolution. The suggested paradigm initiates the NASA software defect lifecycle's 13 distinct steps, demonstrating improved performance. K.K. San and coauthors [9] introduced a novel SRGM method that employs features from completed projects to predict bugs in ongoing projects. Their approach trains similar initiatives to enhance bug identification and employs an RNN-based DLSTM model for prediction.

C. Guo and et al., [10] presented a novel software reliability growth model focused on determining the severity levels of prevalent problems. These levels are analyzed to gauge fault severity and visualized using a logistic curve. Y. Liu and coauthors [11] developed a framework for fault elimination to enhance the accuracy of time delay approaches for software dependability using half-grouped datasets.

In their research, J. Yang and their team [12] considered the incorporation of delays in fault repair time using a time delay model when analyzing the failure processes across various software versions. M. Cinque and their collaborators [13] introduced the Debugging-Workflow-Aware Software Reliability Growth Approach (DWA-SRGM), which is specifically designed to address bugs within software projects. Meanwhile, H. Okamura and colleagues [14] presented an enhanced framework that establishes the correlation between fault discovery times and the durations required for remediation. Finally, H. Sukhwani and coauthors [15] unveiled an SRGM tailored for flight management software utilized in space missions, demonstrating superior performance in real-time software applications.

ORDER STATISTICS

Order statistics involve arranging a sample of random variables in ascending order. Put simply, when you organize observations from a random variable from smallest to largest, you obtain order statistics. These statistics offer insights into the distribution of the random variable and find application in diverse statistical analyses. Denoted as "X_{(1)}, X_{(2)}, ..., X_{(n)}," order statistics are derived from a sample of size n from a random variable X (consisting of observations x1, x2, ..., xn). Here, X_{(1)} is the smallest, X_{(2)} is the second smallest, and X_{(n)} is the largest observation.

Software failure processes are evaluated using failure control principles grounded in inter-failure data accumulation. Following data transformation, failure data is segmented into 4 and 5 cumulative intervals. The time intervals between successive failures illustrate the failure time data. Typically, these failure time data are sorted into non-overlapping subgroups of size 4 or 5 and summed when waiting time for failure isn't critical.

Consider grouping 200 inter-failure times into 40 disjoint subgroups, each comprising 5 observations. The sum of each subgroup represents the interval between every fifth failure, also known as the fifth order statistics when dealing with five observations in a sample.

In our research, we employed the Pareto Type IV model for r = 4 and r = 5,

respectively. The parameters and corresponding values, computed using an iterative technique for cumulative time between failures data, are denoted as the maximum likelihood estimates 'a', 'b,' and 'c.' These estimates facilitate the calculation of m(t).

ILLLUSTRATING THE MLE

A. *Pareto IV distribution Model*

The research suggests estimating software reliability based on order statistics and a Pareto type IV distribution model. It is often used in various fields such as economics, finance, and engineering to model heavy-tailed phenomena where extreme values occur more frequently than in a normal distribution. The mean value function and intensity function for the Pareto type IV NHPP model are defined as follows. The formula for the Cumulative Distribution Function is expressed as:

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

$$= a\,F(t)$$

B. *Mathematical Derivation for Parameter Estimation*

We employ an order statistics technique to derive expressions for estimating the parameters of the Pareto type IV model based on time-domain data. Parameter estimation is a crucial step in anticipating software reliability. Maximum Likelihood Estimation (MLE) aims to determine the parameter values that maximize the likelihood or log-likelihood function. In mathematical terms, this entails finding the values that satisfy the condition $\partial(\log - likelihood)/\partial(parameters) = 0$.

This optimization process is typically carried out using algorithms like gradient descent or Newton's method.

The primary objective of Maximum Likelihood parameter estimation is to select the parameters that maximize the likelihood (probability) of the observed sample data. In essence, MLE approaches are versatile and can be applied to a wide array of models and data sources.

The mean value function of the Pareto type IV model is defined as follows:

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

(1)

Through maximum likelihood (ML) estimation, we determine the values of the parameters a, b, and c. To segment the time domain data into distinct, non-overlapping sequential subgroups of size r, we exponentiate m(t) by raising it to the power of r.

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

(2)

To obtain estimates for the 'n' units, we must initially establish the likelihood function.

The model proposed in this study involves three constants, namely 'a,' 'b,' and 'c,' which are integral components of the mean value function.

By taking the derivative of Eq. (2) concerning 't,' we acquire the following expression:

$$m'(t) = r\left[a\left(1-\frac{1}{\left(1+\frac{t}{c}\right)^b}\right)\right]^{r-1} \frac{ab}{c\left(1+\frac{t}{c}\right)^{b+1}}$$

(3)

The Likelihood function L can be written as

$$L = e^{-m(t)}\prod_{i=1}^{n} m^1(t_i)$$

(4)

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

By substituting Eq. (1) and Eq. (3) into Eq. (4), we obtain the following result:

$$Log\,L = -a^r\left[1-\frac{1}{\left(1+\frac{t}{c}\right)^b}\right]^r + \sum_{i=1}^{n}\log r + \sum_{i=1}^{n}(r-1)\log\left(a-\frac{a}{\left(1+\frac{t}{c}\right)^b}\right) + \sum_{i=1}^{n}\left(\log a+\log b-\log c-(b+1)\log\left(1+\frac{t}{c}\right)\right)$$

Calculate the derivative of Log L with respect to 'a' and set it equal to zero (i.e., $\frac{\partial LogL}{\partial a}=0$)

$$\frac{\partial \log L}{\partial a} = -r\,a^{r-1}\left[1-\frac{1}{\left(1+\frac{t}{c}\right)^b}\right]^r + \sum_{i=1}^{n}(r-1)\left[\frac{1}{a-\frac{a}{\left(1+\frac{t}{c}\right)^b}}\right]\frac{\partial}{\partial a}a\left[1-\frac{1}{\left(1+\frac{t}{c}\right)^b}\right]+\frac{n}{a}$$

$$\frac{\partial LogL}{\partial a}=0$$

$$\therefore a^r = n\left[\frac{(t+c)^b}{(t+c)^b-c^b}\right]^r$$

(5)

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

$$LogL = -a^r\left[1-\frac{1}{\left(1+\frac{t}{c}\right)^b}\right]^r + \sum_{i=1}^{n}\log r + \sum_{i=1}^{n}(r-1)\log\left(a-\frac{a}{\left(1+\frac{t}{c}\right)^b}\right) + \sum_{i=1}^{n}\left(\log a+\log b-\log c-(b+1)\log\left(1+\frac{t}{c}\right)\right)$$

Calculate the derivative of Log L with respect to 'b' and set it equal to zero.

$$g(b)=\left[\left(\frac{nr}{(t+1)^b-1}\right)\log\left(\frac{1}{1+t}\right)\right]-\sum_{i=1}^{n}\frac{r-1}{(1+t_i)^b-1}\log\left(\frac{1}{1+t_i}\right)+\frac{n}{b}-\sum_{i=1}^{n}\log(1+t_i)$$

(6)

$$g^1(b)=-nr\log\left(\frac{1}{1+t}\right)\frac{(t+1)^b\log(t+1)}{\left[(t+1)^b-1\right]^2} + \sum_{i=1}^{n}(r-1)\log\left(\frac{1}{1+t_i}\right)\frac{(t_i+1)^b\log(1+t_i)}{\left[(1+t_i)^b-1\right]^2}-\frac{n}{b^2}$$

(7)

Calculate the derivative of Log L with respect to 'b' and set it equal to zero.

(i.e., $\frac{\partial LogL}{\partial c}=0$. we get

$$g(c)=\frac{nr}{(t+c)}+\sum_{i=1}^{n}(r-1)\left(\frac{-1}{t_i+c}\right)-\frac{n}{c}+\sum_{i=1}^{n}\frac{2t}{\left(1+\frac{t_i}{c}\right)c^2}$$

(8)

$$g^1(c)=\frac{-nr}{(t+c)^2}+\sum_{i=1}^{n}(r-1)\frac{1}{(t_i+c)^2}-\frac{n}{c^2}+\sum_{i=1}^{n}\frac{-4t_i^2}{(t_i+c)^2c^3}$$

(9)

C. Estimated parameters and their control limits

By utilizing inter-failure time data along with maximum likelihood estimation, the parameters were calculated. The process is deemed to be out of control under two conditions: when the time to observe a single failure is either below the Lower Control Limit (LCL) or above the Upper Control Limit (UCL), as specified by the control chart's limits. Our primary objectives involve monitoring the failure process and detecting changes in the intensity parameter. Such occurrences are considered false alarms when the process is operating normally. This scenario is denoted as a false alarm, and although other false alarm probability values are feasible, the conventional setting is at 0.27%. It's crucial for the permissible false alarm probability to be determined based on the actual product or process.

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Control limits can be calculated by

$T_u = 0.99865$

$T_c = 0.5$

$T_l = 0.00135$

Table 1 displays the estimated parameters and corresponding control limits for the Failure Count Chart (FCC), with a false alarm risk set at 0.0027, applied to both the Musa and SYS2 datasets. The control limits are calculated using these estimated parameters. These limits play a crucial role in assessing whether the software process is within control or not. The table provides the estimated values of 'a,' 'b,' and 'c,' along with their respective control limits, for both 4th-order and 5th-order statistics.

Table 1. Estimates of the parameters and the control limits for orders 4 and 5

Dataset	Order	Estimates			Control Limits		
		'a'	'b'	'c'	"UCL"	"CL"	"LCL"
Musa	4	3.407049	0.110178	1.217387	3.402449	1.703524	0.004599
	5	2.724650	0.110720	1.197433	2.720971	0.003678	1.362325
Sys2	4	3.187556	0.098599	1.217576	3.183252	0.004303	1.593778
	5	2.626644	0.098197	1.197625	2.623098	1.313322	0.003545

Distribution of Time between Failures

Tables 2 to 5 present calculations of the mean differences in the rth order cumulative time between failures data for the designated datasets. These computations serve as the basis for generating Figures 1 to 4, where the x-axis represents the failure numbers, and the y-axis displays the mean differences between successive data points. Additionally, control limits are superimposed on the Failure Control Chart. The appearance of a point beyond these control limits triggers an alarm. Points situated above the control limit indicate enhanced quality, whereas points residing within the control limits signify stability in the software process.

Table 2: Consecutive variances of 4th order mean values of Musa

Failure No	Fourth order cumulatives	m(t)	Consecutive variances	Failure No	Fourth order cumulatives	m(t)	Consecutive variances
1	227	1.493002	0.135864	18	16358	2.211612	0.014591
2	444	1.628866	0.101794	19	18287	2.226203	0.015187
3	759	1.730660	0.059819	20	20567	2.241390	0.020323
4	1056	1.790480	0.108586	21	24127	2.261713	0.020653
5	1986	1.899065	0.048715	22	28460	2.282366	0.015982
6	2676	1.947780	0.078945	23	32408	2.298348	0.018176
7	4434	2.026725	0.020790	24	37654	2.316524	0.013088
8	5089	2.047515	0.008551	25	42015	2.329612	0.000791

9	5389	2.056066	0.024890	26	42296	2.330403	0.015621
10	6380	2.080956	0.022399	27	48296	2.346024	0.008697
11	7447	2.103355	0.008850	28	52042	2.354721	0.003075
12	7922	2.112205	0.036342	29	53443	2.357796	0.006380
13	10258	2.148546	0.011815	30	56485	2.364176	0.011836
14	11175	2.160361	0.015933	31	62651	2.376013	0.003986
15	12559	2.176295	0.009618	32	64893	2.379999	0.017807
16	13486	2.185913	0.016661	33	76057	2.397806	0.016934
17	15277	2.202573	0.009038	34	88683	2.414740	

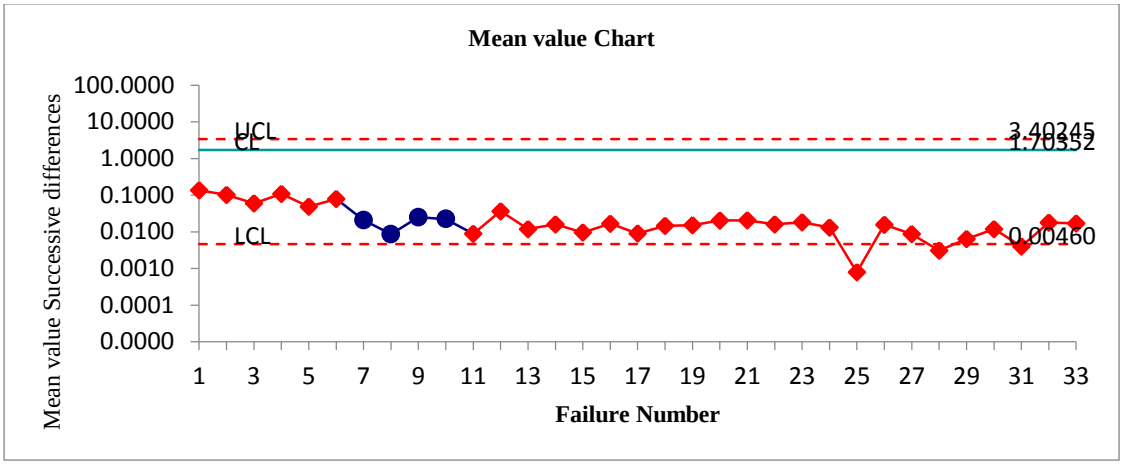


Fig 1: FCC for Musa dataset of order 4

Table 3: Consecutive variances of 5th order mean value of Musa dataset

Failure No	Fifth order cumulatives	m(t)	Consecutive variances	Failure No	Fifth order cumulatives	m(t)	Consecutive variances
1	342	1.268389	0.080132	15	17758	1.783891	0.015172
2	571	1.348521	0.077996	16	20567	1.799062	0.023366
3	968	1.426517	0.099204	17	25910	1.822428	0.012404
4	1986	1.525721	0.057567	18	29361	1.834832	0.024143
5	3098	1.583288	0.060067	19	37642	1.858975	0.010470
6	5049	1.643354	0.006329	20	42015	1.869445	0.007318
7	5324	1.649684	0.021317	21	45406	1.876763	0.007908
8	6380	1.671001	0.020873	22	49416	1.884671	0.007044
9	7644	1.691875	0.031248	23	53321	1.891714	0.005299
10	10089	1.723122	0.009360	24	56485	1.897013	0.009454
11	10982	1.732482	0.014630	25	62661	1.906467	0.015365
12	12559	1.747112	0.016946	26	74364	1.921833	0.011346

13	14708	1.764057	0.010123	27	84566	1.933179	
14	16185	1.774181	0.009710				

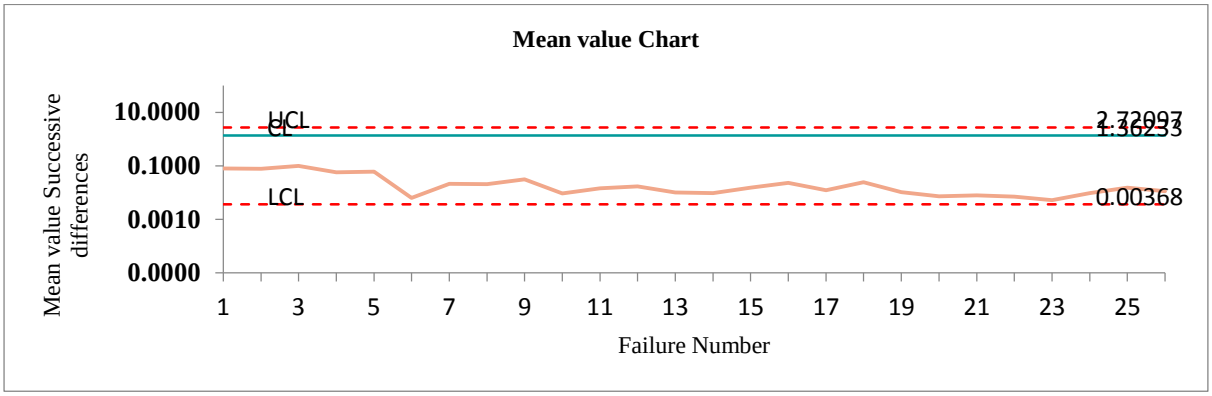


Fig 2: FCC for Musa dataset of order 5

Table 4: Consecutive variances of 4th order mean values of Sys2 dataset

Failure No	Fourth order cumulatives	m(t)	Consecutive variances	Failure No	Fourth order cumulatives	m(t)	Consecutive variances
1	1576	1.615099	0.143071	12	34467	2.027437	0.018999
2	4149	1.758170	0.047063	13	40751	2.046436	0.018875
3	5827	1.805233	0.072588	14	48262	2.065311	0.010775
4	10071	1.877821	0.020687	15	53223	2.076085	0.005871
5	11836	1.898507	0.032053	16	56160	2.081956	0.009971
6	15280	1.930560	0.012136	17	61565	2.091928	0.013501
7	16860	1.942696	0.018173	18	69815	2.105429	0.018076
8	19572	1.960868	0.023563	19	82822	2.123504	0.010050
9	23827	1.984431	0.020058	20	91190	2.133554	0.007140
10	28257	2.004489	0.014010	21	97698	2.140694	
11	31886	2.018499	0.008937				

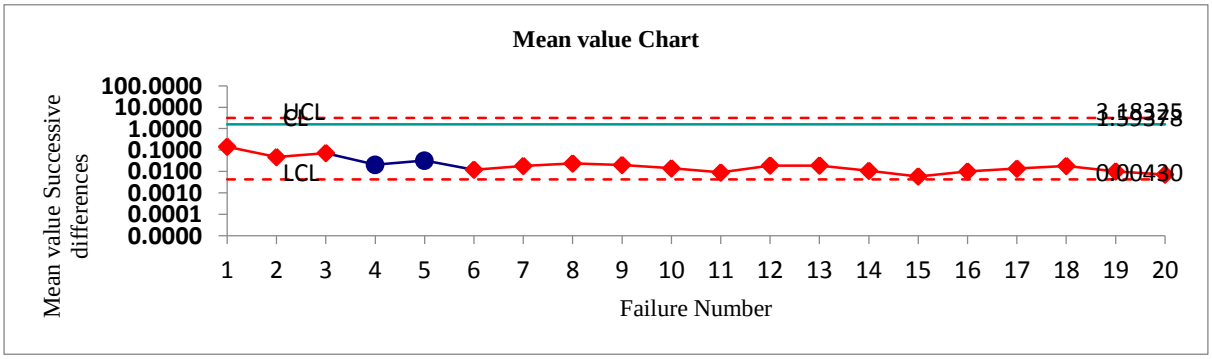


Fig 3: FCC for Sys2 dataset of order 4

Table 5: Consecutive variances of 5th order mean values of Sys2 dataset

Failure No	Fifth order cumulatives	m(t)	Consecutive variances	Failure No	Fifth order cumulatives	m(t)	Consecutive variances
1	2610	1.391923	0.062641	10	39856	1.681852	0.013499

2	4436	1.454565	0.068118	11	46147	1.695352	0.012955
3	8163	1.522683	0.039546	12	53223	1.708306	0.009239
4	11836	1.562228	0.029024	13	58996	1.717546	0.011777
5	15685	1.591252	0.013874	14	67374	1.729323	0.015123
6	17995	1.605126	0.020963	15	80106	1.744446	0.011155
7	22226	1.626089	0.023311	16	91190	1.755601	0.006736
8	28257	1.649400	0.012883	17	98692	1.762337	
9	32346	1.662283	0.019569				

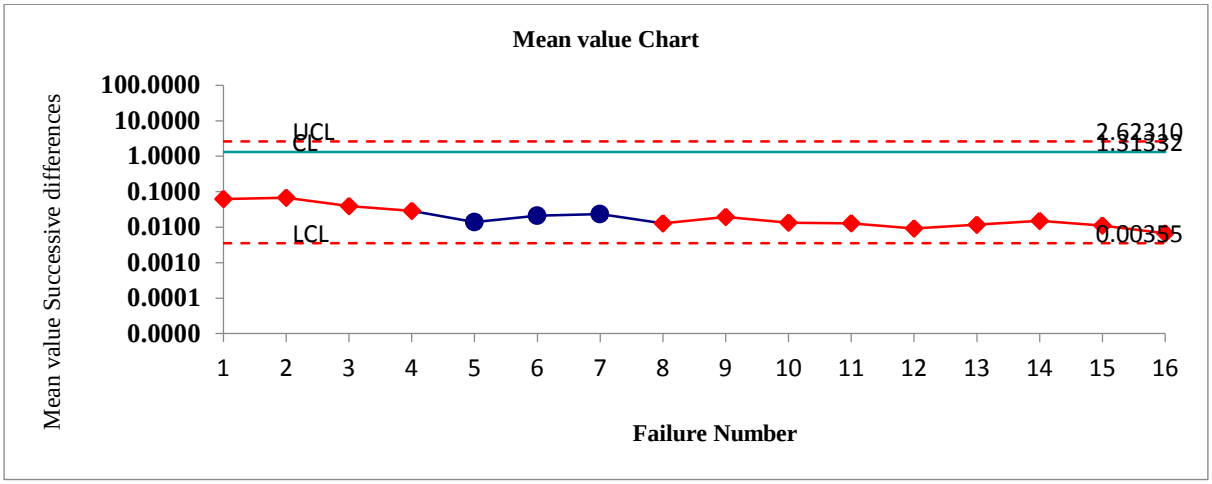


Fig. 4. FCC for Sys2 dataset of order 5

CONCLUSION

We've plotted the 4th and 5th order failure counts against the serial order of failures using the estimated mean value function. Parameters were estimated using the Maximum Likelihood Estimation (MLE) approach. The successive differences in the Sys2 dataset show moderate fluctuations within the control limits, whereas the successive differences in the Musa dataset have exceeded the control limits. As a result, we can confidently assert that our estimation method and the control chart provide strong support for their applicability in identifying optimal control processes or detecting noteworthy out-of-control signals.

REFERENCES

1. Ullah, Najeeb & Morisio, Maurizio & Vetro, Antonio. (2013). A Comparative Analysis of Software Reliability Growth Models using Defects Data of Closed and Open Source Software. Proceedings of the 2012 IEEE 35th Software Engineering Workshop, SEW 2012. 187-192. 10.1109/SEW.2012.26.
2. Kapil Sharma, et al, "Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach", IEEE Transactions On Reliability, VOL. 59, NO. 2, JUNE 2010.
3. Bruno Rossi et al, "Modelling Failures Occurrences of Open Source Software with Reliability Growth", journal of Open Source Software: New Horizons, page 268-280, 2010.
4. Cobra Rahmani et al, "A Comparative Analysis of Open Source Software Reliability", Journal of Software, page 1384-1394, 2010.
5. Gutta Sridevi, R.Satya Prasad and K.V.Murali Mohan, "Monitoring Burr Type XII Software Quality Using SPC",International Journal of Applied Engineering Research (IJAER), Vol. 9,

No. 22 (2014), pp:16651- 16660, ISSN:0973-4462.

6. K. S. Kumari, B. Amulya and R. S. Prasad, "Comparative study of Pareto Type II with HLD in assessing the software reliability with order statistics approach using SPC," 2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014], 2014, pp. 1630-1636, doi: 10.1109/ICCPCT.2014.7054824.
7. Y. Wu, Y. Zhang and M. Lu, "Software reliability accelerated testing method based on mixed testing," 2010 Proceedings - Annual Reliability and Maintainability Symposium (RAMS), 2010, pp. 1-6, doi: 10.1109/RAMS.2010.5448017.
8. M. Nafreen, M. Luperon, L. Fiondella, V. Nagaraju, Y. Shi and T. Wand ji, "Connecting Software Reliability Growth Models to Software Defect Tracking," 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), 2020, pp. 138-147, doi: 10.1109/ISSRE5003.2020.00022.
9. K. K. San, H. Washizaki, Y. Fukazawa, K. Honda, M. Taga and A. Matsuzaki, "DC-SRGM: Deep Cross-Project Software Reliability Growth Model," 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2019, pp. 61-66, doi: 10.1109/ISSREW.2019.00044.
10. C. Guo, S. Zhou, J. Li, F. Chen, D. Li and X. Huang, "A Novel Software Reliability Growth Model of Safety-critical Software Considering Fault Severity Classification," 2019 4th International Conference on System Reliability and Safety (ICSRS), 2019, pp. 25-29, doi: 10.1109/ICSRS48664.2019.8987594
11. Y Liu, M. Xie, J. Yang and M. Zhao, "A new framework and application of software reliability estimation based on fault detection and correction processes", Proc. IEEE International Conference on Software Quality Reliability and Security, pp. 65-74, 2015.
12. J. Yang, Y Liu, M. Xie and M. Zhao, "Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes", Journal of Systems and Software, vol. 115, pp. 102-110, 2016.
13. M. Cinque, D. Cotroneo, A. Pecchia, R. Pietrantuono and S. Russo, "Debugging-workflowaware software reliability growth analysis", Software Testing Verification and Reliability, vol. 27, no. 7, pp. e1638, 2017.
14. H. Okamura and T. Dohi, "A generalized bivariate modeling framework of fault detection and correction processes", Proc. IEEE International Symposium on Software Reliability Engineering, pp. 35-45, 2017.
15. H. Sukhwani, J. Alonso, K. Trivedi and I. Mcginnis, "Software reliability analysis of NASA space flight software: A practical experience", Proc. IEEE International Conference on Software Quality Reliability and Security, pp. 386-397, 2016.