Software Reliability: An Order Statistics Approach using SPC

N.Supriya
Dept. Of Computer Science, Adikavi Nannaya University, Rajahmundry, Andhrapradesh, India.

Dr. R. Satya Prasad
Associate Professor, Dept. of Computer Science & Engg., Acharya Nagrjuna University Guntur, Andhrapradesh, India.

G.Krishna Mohan
Reader, Dept. Of Computer Science, P.B.Siddhartha College Vijayawada, Andhrapradesh, India.

ABSTRACT
Control charts are widely used for process monitoring. Software reliability process can be monitored efficiently by using Statistical Process Control (SPC). It assists the software development team to identify failures and actions to be taken during software failure process and hence, assures better software reliability. In this paper we proposed a control mechanism based on order statistics of the cumulative quantity between observations of time domain failure data using mean value function of Exponential imperfect debugging, which is based on Non Homogenous Poisson Process (NHPP). The Maximum Likelihood Estimation (MLE) method is used to derive the point estimators of the distribution.

Keywords
Statistical Process Control, Exponential imperfect debugging, Mean Value function, Probability limits, Control Charts.

1. INTRODUCTION
Software Reliability is the application of statistical techniques to data collected during system development and operation to specify, predict, estimate, and assess the reliability of software-based systems. To identify and eliminate errors in software development process and also to improve software reliability, the Statistical Process Control concepts and methods are the best choice. SPC concepts and methods are used to monitor the performance of a software process over time in order to verify that the process remains in the state of statistical control. It helps in finding assignable causes, long term improvements in the software process. Software quality and reliability can be achieved by eliminating the causes or improving the software process or its operating procedures [2].

The most popular technique for maintaining process control is control charting. Software process control is used to secure the quality of the final product which will conform to predefined standards. In any process, regardless of how carefully it is maintained, a certain amount of natural variability will always exist. A process is said to be statistically "in-control" when it operates with chance causes of variation. On the other hand, when assignable causes are present, the process is statistically "out-of-control". The control charts can be classified into several categories, as per several distinct criteria. Control charts should be capable to create an alarm when a shift in the level of one or more parameters of the underlying distribution or a non-random behavior occurs. Normally, such a situation will be reflected in the control chart by points plotted outside the control limits or by the presence of specific patterns. The most common non-random patterns are cycles, trends, mixtures and stratification [3]. For a process to be in control the control chart should not have any trend or nonrandom pattern.

Statistical Process Control (SPC) is about using control charts to manage software development efforts, in order to effect software process improvement. The practitioner of SPC tracks the variability of the process to be controlled. The early detection of software failures will improve the software reliability. The selection of proper SPC charts is essential to effective statistical process control implementation and use. The SPC chart selection is based on data, situation and need [4]. Many factors influence the process, resulting in variability. The causes of process variability can be broadly classified into two categories, viz., assignable causes and chance causes.control

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 in control, 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 [10].

2. LITERATURE SURVEY
This section presents the theory that underlies exponential imperfect distribution and maximum likelihood estimation for complete data. If 't' is a continuous random variable with pdf: $f(t;\theta_1,\theta_2,\ldots,\theta_k)$. Where $\theta_1,\theta_2,\ldots,\theta_k$ 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 expected number of 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 distribution function. The failure intensity function $\lambda(t)$ in case of the finite failure NHPP models is given by: $\lambda(t)=aF'(t)$ [9].

2.1 NHPP model
The Non-Homogenous Poisson Process (NHPP) based software reliability growth models (SRGMs) are proved to be quite successful in practical software reliability engineering [5]. The main issue in the NHPP model is to determine an appropriate mean value function to denote the expected number of failures

experienced up to a certain time point. Model parameters can be estimated by using Maximum Likelihood Estimate (MLE). Various NHPP SRGMs have been built upon various assumptions. Many of the SRGMs assume that each time a failure occurs, the fault that caused it can be immediately removed and no new faults are introduced. Which is usually called perfect debugging. Imperfect debugging models have proposed a relaxation of the above assumption [6,7].

Let $\{N(t), t \geq 0\}$ be the cumulative number of software failures by time 't'. m(t) is the mean value function, representing the expected number of software failures by time 't'. $\lambda(t)$ is the failure intensity function, which is proportional to the residual fault content. Thus $m(t) = a(1-e^{-bt})$ and $\lambda(t) = \frac{dm(t)}{dt} = b(a - m(t))$. where 'a' denotes the initial number of faults contained in a program and 'b' represents the fault detection rate. In software reliability, the initial number of faults and the fault detection rate are always unknown. The maximum likelihood technique can be used to evaluate the unknown parameters. In general NHPP SRGM $\lambda(t)$ can be expressed as $\lambda(t) = \frac{dm(t)}{dt} = b(t)\left[a(t) - m(t)\right]$. where $a(t)$ is the time-dependent fault content function which includes the initial and introduced faults in the program and $b(t)$ is the time-dependent fault detection rate.

A constant $a(t)$ implies the perfect debugging assumption, i.e no new faults are introduced during the debugging process. If we assume that, when the detected faults are removed, then there is a possibility to introduce new faults with a constant rate 'β'. Then the mean value function [2] is given as $m(t) = \frac{a}{1-\beta}\left[1 - e^{-(1-\beta)bt}\right]$.

2.2 Order Statistics

Order statistics deals with properties and applications of ordered random variables and of functions of these variables. The use of order statistics is significant when failures are frequent or inter failure time is less. Let X denote a continuous random variable with probability density function (pdf) f(x) and cumulative distribution function (cdf) F(x), and let (X1 , X2 , …, Xn) denote a random sample of size n drawn on X. The original sample observations may be unordered with respect to magnitude. A transformation is required to produce a corresponding ordered sample. Let (X(1) , X(2) , …, X(n)) denote the ordered random sample such that X(1) < X(2) < … < X(n); then (X(1), X(2), …, X(n)) are collectively known as the order statistics derived from the parent X. The various distributional characteristics can be known from Balakrishnan and Cohen [1].

2.3 Exponential imperfect debugging distribution

Mean Value Function of the distribution is $m(t) = \frac{a}{1-\beta}\left[1 - e^{-(1-\beta)bt}\right]$. For rth order statistics, the mean value function is expressed as $m^r(t) = \left\{\frac{a}{1-\beta}\left(1 - e^{-(1-\beta)bt}\right)\right\}^r$. The failure intensity function is given as: $\left[m^r(t)\right]' = \lambda^r(t) = \frac{r.b.a^r}{(1-\beta)^{r-1}}\left\{1 - e^{-(1-\beta)bt}\right\}^{r-1} . e^{-(1-\beta)bt}$.

2.4 MLE (Maximum Likelihood) Parameter Estimation

The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability (likelihood) of the sample data. The method of maximum likelihood is considered to be more robust (with some exceptions) and yields estimators with good statistical properties. In other words, MLE methods are versatile and apply to many models and to different types of data. Although the methodology for maximum likelihood estimation is simple, the implementation is mathematically intense. Using today's computer power, however, mathematical complexity is not a big obstacle. If we conduct an experiment and obtain N independent observations, $t_1, t_2, \ldots, t_N$. Then the likelihood function [8] may be given by the following product:

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

ikelihood function by using λ(t) is:L = $\prod_{i=1}^{n} \lambda(t_i)$

The logarithmic likelihood function is given by: $\Lambda = \ln L = \sum_{i=1}^{N} \ln f(t_i; \theta_1, \theta_2, \ldots, \theta_k)$

Log Likelihood function is:Log L = log ($\prod_{i=1}^{n} \lambda(t_i)$)

The maximum likelihood estimators (MLE) of $\theta_1, \theta_2, \ldots, \theta_k$ are obtained by maximizing L or Λ, where Λ is ln L . By maximizing Λ, which is much easier to work with than L, the maximum likelihood estimators (MLE) of $\theta_1, \theta_2, \ldots, \theta_k$ are the simultaneous solutions of k equations such as: $\frac{\partial(\Lambda)}{\partial\theta_j} = 0$, j=1,2,…,k

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

3. ILLUSTRATING THE METHOD

3.1 Parameter estimation

To estimate 'a' and 'b' , for a sample of n units (all tested to failure), first obtain the likelihood function: assuming $\beta = 0.05$.

The Likelihood function $L = \prod_{i=1}^{n} \lambda^r (t_i)$

Take the natural logarithm on both sides, The Log Likelihood function is given as:

Log L = $\log[\prod_{i=1}^{n} \lambda^r (t_i)]$

$$\log L = \log\left[\prod_{i=1}^{n} \frac{r.b.a^r}{(1-\beta)^{r-1}} \left\{1-e^{-(1-\beta)bt}\right\}^{r-1} .e^{-(1-\beta)bt}\right]$$

$$\log L = \sum_{i=1}^{n} \log\left\{\lambda^r (t_i)\right\} - m^r (t_n)$$

$$\log L = \sum_{i=1}^{n} \log\left(\frac{rba^r}{(1-\beta)^{r-1}} \left\{1-e^{-(1-\beta)bt_i}\right\}^{r-1} .e^{-(1-\beta)bt_i}\right) - \left\{\frac{a}{1-\beta}\left(1-e^{-(1-\beta)bt_n}\right)\right\}^r$$

Partially differentiating w.r.t 'a' and equating to 0.(i.e. $\frac{\partial \log L}{\partial a} = 0$).

$$a^r = \frac{n(1-\beta)^r}{\left(1-e^{-(1-\beta)bt_n}\right)^r}$$

Partially differentiating w.r.t 'b' and equating to 0.(i.e. $\frac{\partial \log L}{\partial b} = 0$).

$$g(b) = \sum_{i=1}^{n}\left\{\frac{1}{b} + \frac{(1-\beta)(r-1)t_i}{\left(e^{(1-\beta)bt_i}-1\right)} - (1-\beta)t_i\right\} - \frac{n.r(1-\beta)t_n}{\left(e^{(1-\beta)bt_n}-1\right)}$$

.

Again Partially differentiating w.r.t 'b' and equating to 0.(i.e. $\frac{\partial^2 \log L}{\partial b^2} = 0$).

$$g'(b) = -\frac{n}{b^2} - (1-\beta)^2 (r-1)\sum_{i=1}^{n} \frac{t_i e^{(1-\beta)bt_i}}{\left(e^{(1-\beta)bt_i}-1\right)^2} + \frac{nr(1-\beta)^2 t_n^2 e^{(1-\beta)bt_n}}{\left(e^{(1-\beta)bt_n}-1\right)^2}$$

The parameter 'b' is estimated by iterative Newton Raphson Method using $b_{n+1} = b_n - \frac{g(b_n)}{g'(b_n)}$, which is substituted in finding 'a'.

3.2 Distribution of Time between failures

We compute the software failures process through Mean Value Control chart based on the inter failure data given in Table 1. We used cumulative time between failures data which is ordered through a transformation for software reliability monitoring using exponential imperfect distribution. The transformation being applied is, the failure data is made into groups of 4, 5 and then cumulated. The inter failure time data represent the time laps between every two consecutive failures. On the other hand if a reasonable waiting time for failures is not a serious problem. We can group the inter failure time data into non overlapping successive subgroups of size 4 or 5 and add the failures times with needs of groups. For instance if a data of 100 inter failure times are available, we can group them into 20 disjoint subgroups of size 5. The sum totals in each subgroup would represent the time laps between every 5th failure. In the theory of statistics such a subtotal is defined as the 5th order statistics in a sample of size 5.

In general for inter failure data of size 'n' if 'r' is any natural number less than n and preferably a factor of 'n' we can conveniently divide the data into 'k' disjoint subgroups(k=n/r) and the cumulative total meets subgroup indicate the time between every rth failure. The probability distribution of such a time laps would be better in the rth order statistic in a subgroup of size 'r'. which would be equal to the rth power of the distribution function of the original variable.

The parameters of the mean value function with the revised distribution function would determine the control limits of a new control chart involving order statistics. Hence they need a separate study. In the present paper we have taken r = 4, 5 and the basic distribution as exponential imperfect choice of r beyond 5 may create an unduly long waiting time for the occurrence of every rth failure. '$\hat{a}$' and '$\hat{b}$' are Maximum Likely hood Estimates (MLEs) of parameters and the values can be computed using iterative method for the given cumulative time between failures data [9] shown in table 1. The data is documented in Lyu(1996). There are in total 136 faults reported and the time between failures in seconds. Using 'a' and 'b' values we can compute $m(t)$.

Table:1 Time between failures of a software

F.No	TBF(h)	F.No	TBF(h)	F.No	TBF(h)	F.No	TBF(h)
1	3	35	227	69	529	103	108
2	30	36	65	70	379	104	0
3	113	37	176	71	44	105	3110
4	81	38	58	72	129	106	1247
5	115	39	457	73	810	107	943
6	9	40	300	74	290	108	700
7	2	41	97	75	300	109	875
8	91	42	263	76	529	110	245
9	112	43	452	77	281	111	729
10	15	44	255	78	160	112	1897
11	138	45	197	79	828	113	447
12	50	46	193	80	1011	114	386
13	77	47	6	81	445	115	446
14	24	48	79	82	296	116	122
15	108	49	816	83	1755	117	990
16	88	50	1351	84	1064	118	948
17	670	51	148	85	1783	119	1082
18	120	52	21	86	860	120	22
19	26	53	233	87	983	121	75
20	114	54	134	88	707	122	482
21	325	55	357	89	33	123	5509
22	55	56	193	90	868	124	100
23	242	57	236	91	724	125	10
24	68	58	31	92	2323	126	1071
25	422	59	369	93	2930	127	371
26	180	60	748	94	1461	128	790
27	10	61	0	95	843	129	6150
28	1146	62	232	96	12	130	3321
29	600	63	330	97	261	131	1045
30	15	64	365	98	1800	132	648
31	36	65	1222	99	865	133	5485
32	4	66	543	100	1435	134	1160
33	0	67	10	101	30	135	1864
34	8	68	16	102	143	136	4116

Assuming an acceptable probability of false alarm of 0.27%, the control limits can be obtained as [10]:

$$T_U = 1 - e^{-(bt)^{\beta}} = 0.99865$$

$$T_C = 1 - e^{-(bt)^{\beta}} = 0.5$$

$$T_L = 1 - e^{-(bt)^{\beta}} = 0.00135$$

These limits are converted to $m(t_U)$, $m(t_C)$ and $m(t_L)$ form respectively. They are used to find whether the software process is in control or not by placing the points in Mean value chart shown in figure 1 & 2. A point below the control limit $m(t_L)$ indicates an alarming signal. A point above the control limit $m(t_U)$ indicates better quality. If the points are falling within the control limits, it indicates the software process is in stable condition. The estimated values of 'a' and 'b' and their control limits for both 4th-order and 5th-order statistics are as follows.

Table: 2 Successive differences of 4 order mean values

F. No	4-order Cumulative	m(t)	SD
1	227	0.028252	0.026702
2	444	0.054953	0.038235
3	759	0.093189	0.035489
4	1056	0.128677	0.107693
5	1986	0.236370	0.076646
6	2676	0.313015	0.183447
7	4434	0.496463	0.064228
8	5089	0.560691	0.028705
9	5389	0.589396	0.091742
10	6380	0.681138	0.093700
11	7447	0.774837	0.040091
12	7922	0.814928	0.183553
13	10258	0.998482	0.066252
14	11175	1.064734	0.094273
15	12559	1.159007	0.059503
16	13486	1.218510	0.107247
17	15277	1.325757	0.060121
18	16358	1.385878	0.099340
19	18287	1.485218	0.105417
20	20567	1.590635	0.141816
21	24127	1.732452	0.141068
22	28460	1.873520	0.103901
23	32408	1.977421	0.109199
24	37654	2.086620	0.070885
25	42015	2.157505	0.004046
26	42296	2.161552	0.073833
27	48296	2.235385	0.035814
28	52042	2.271199	0.011722
29	53443	2.282921	0.022729
30	56485	2.305650	0.036494
31	62651	2.342145	0.010657
32	64893	2.352802	0.038120
33	76057	2.390922	0.023508
34	88682	2.414429	

Table: 3 Successive differences of 5 order mean values

F. No	5-order Cumul	m(t)	SD
1	342	0.034070	0.022481
2	571	0.056551	0.038353
3	968	0.094905	0.094851
4	1986	0.189756	0.098103
5	3098	0.287859	0.159160
6	5049	0.447020	0.021180
7	5324	0.468201	0.078612
8	6380	0.546813	0.088665
9	7644	0.635479	0.156018
10	10089	0.791497	0.052277
11	10982	0.843775	0.086669
12	12559	0.930444	0.107365
13	14708	1.037810	0.067220
14	16185	1.105030	0.066204
15	17758	1.171234	0.105719
16	20567	1.276953	0.163515
17	25910	1.440469	0.084132
18	29361	1.524601	0.150350
19	37642	1.674952	0.057080
20	42015	1.732033	0.036240
21	45406	1.768273	0.035463
22	49416	1.803737	0.028186
23	53321	1.831923	0.019039
24	56485	1.850963	0.029338
25	62661	1.880301	0.035525
26	74364	1.915827	0.017593
27	84566	1.933421	

Table: 4 Parameter estimates and their control limits of 4 and 5 order

Order	a	b	$M(t_u)$	$M(t_c)$	$M(t_l)$
4	2.31822	0.00005	2.31509	1.15911	0.00313
5	1.86105	0.00005	1.85854	0.93052	0.00251

By placing the time between failures of cumulative data shown in tables 2 and 3 on y axis and failure number on x axis and the values of control limits are placed on Failure Control Chart, figure 1 and 2 is obtained respectively. The Mean Value charts shows that the failure data of 4th-order and of 5th -order has fallen between $m(t_L)$ and $m(t_U)$, which indicates the process is under control. The software quality is determined by detecting failures at an early stage.

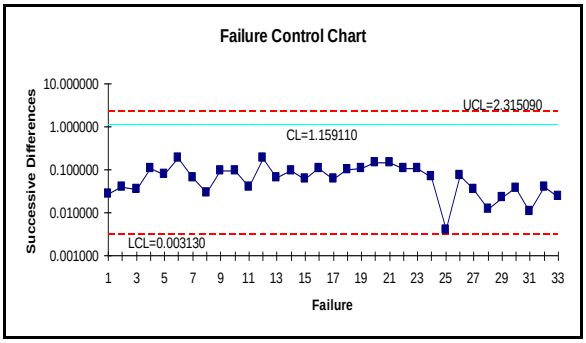

Figure: 1Failure Control Chart

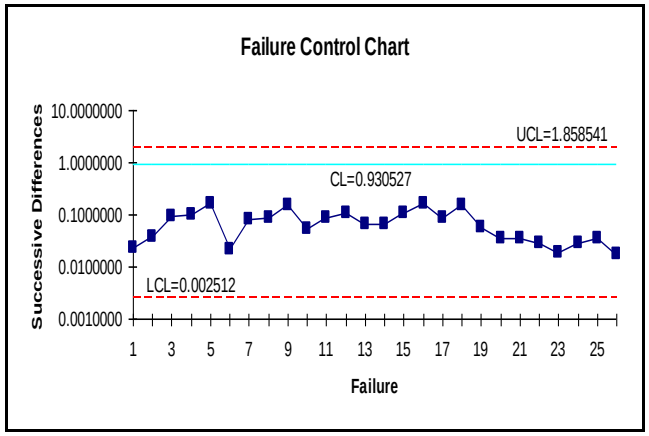

Figure: 2Failure Control Chart

4. CONCLUSION

The failures control charts of Fig 1 and 2 have shown within control signals. i.e below CL and above LCL. By observing failures control charts, this indicates that the debugging process is modified inorder to achieve better quality product. This concept is incorporated in the modified mean value function, this has shown an improvement in the quality of the software by giving within control of the failure time data for all the data sets. Therefore, analysis of failure time data with order statistics approach and imperfect debugging would always resulting within control software product for exponential models.

5. REFERENCES

[1] Balakrishnan. N., Clifford Cohen. A., "Order Statistics and Inference", Academic Press, INC. page 13. 1991.

[2] Kimura, M., Yamada, S., Osaki, S.,"Statistical Software reliability prediction and its applicability based on mean time between failures". Mathematical and Computer Modeling Volume 22, Issues 10-12, 1995. 149-155.

[3] Koutras, M.V., Bersimis, S., Maravelakis,P.E., "Statistical process control using shewart control charts with supplementary Runs rules" Springer Science + Business media 9: 2007. 207-224.

[4] MacGregor, J.F., Kourti, T., "Statistical process control of multivariate processes". Control Engineering Practice Volume 3, Issue 3, March 1995, 403-414.

[5] Musa, J.D., Iannino, A., Okumoto, k., "Software Reliability: Measurement Prediction Application". McGraw-Hill, New York. 1987.

[6] Ohba, M., "Software reliability analysis model". IBM J. Res. Develop. 28, 1984. 428-443.

[7] Pham. H., "Software reliability assessment: Imperfect debugging and multiple failure types in software development". EG&G-RAAM-10737; Idaho National Engineering Laboratory. 1993.

[8] Pham. H., "Handbook of Reliability Engineering", Springer. 2003.

[9] Pham. H., "System software reliability", Springer. 2006.

[10] Swapna S. Gokhale and Kishore S.Trivedi, "Log-Logistic Software Reliability Growth Model". The 3rd IEEE International Symposium on High-Assurance Systems Engineering. IEEE Computer Society. 1998.

[11] Xie, M., Goh. T.N., Ranjan.P., "Some effective control chart procedures for reliability monitoring" -Reliability engineering and System Safety 77¸ 2002. 143 -150.

6. AUTHORS PROFILE

Mrs. N.Supriya Working as Academic Consultant, in the department of Computer Science, Adikavi Nannaya university, Rajahmundry. She is at present pursuing Ph.D at Acharya Nagarjuna University. Her research interests lies in Softwrare Engineering and Data Mining.

Dr. R. Satya Prasad received Ph.D. degree in Computer Science in the faculty of Engineering in 2007 from Acharya Nagarjuna University, Andhra Pradesh. He received gold medal from Acharya Nagarjuna University for his out standing performance in Masters Degree. He is currently working as Associate Professor and H.O.D, in the Department of Computer Science & Engineering, Acharya Nagarjuna University. His current research is focused on Software Engineering. He has published several research papers in National & International Journals.

Mr. G. Krishna Mohan is working as a Reader in the Department of Computer Science, P.B.Siddhartha College, Vijayawada. He obtained his M.C.A degree from Acharya Nagarjuna University in 2000, M.Tech from JNTU, Kakinada, M.Phil from Madurai Kamaraj University and pursuing Ph.D at Acharya Nagarjuna University. His research interests lies in Data Mining and Software Engineering.