CDQM, An Int. J., Volume 17, Number 4, December 2014, pp. 15-26

COMMUNICATIONS
IN
DEPENDABILITY AND
QUALITY
MANAGEMENT
An International Journal

UDC 004.41:519.2
COBISS.SR-ID 213385996

Statistical Process Control and Software Reliability Trend: An Analysis Based on Inter Failure Time Data

Adarsh Anand[1], Navneet Bhatt[1], Deepti Aggrawal[1]
and ***Ljubisa Papic[2]***

[1] Department of Operational Research, University of Delhi, Delhi 110007, India
E-mail: adarsh.anand86@gmail.com, navneetbhatt@live.com, deepti.aggrawal@gmail.com
[2] DQM Research Center, P. O. Box 132, 32102 Cacak, Serbia
E-Mail: dqmcenter@open.telekom.rs

accepted October 14, 2014

Summary
In the last few decades for the critical business applications there has been a requirement of continuous advancement in software industry. For achieving this many software reliability growth models (SRGMs) have been developed focusing on the fault removal process in the software. There is an increased interest for monitoring and improving the software process using control charts during the testing phase. In control chart, analyzing the patterns and establishing control limits can help us to identify problems in the process and these identified faults are predicted using various SRGMs. The SRGMs help us in predicting how the faults are detected and removed during the whole testing process but are insufficient in identifying those faults which requires more testing efforts during the removal phenomenon for the faults. In the present study we utilize the concept of control charts in the software testing process for detecting those out of control faults which requires more testing efforts while removing and results in the declination of reliability. For the validation purpose various trend tests available in the literature have been applied.

Key words: *Non-homogeneous Poisson process (NHPP), Software Reliability Growth Model (SRGM), Statistical Process Control (SPC), Control Chart.*

1. INTRODUCTION

In the competitive environment the role of software is expanding very rapidly, hence the software engineer plays a key role in the reliability, quality and the satisfaction of the customer for the defect free operation of the software. Software testing is a major part of quality control during the

development of software which requires rigorous testing. Once software is up for release in the market it has to be free from defects. Assessment of software reliability is an important task to predict and evaluate the capability and reliability of the software system. During the testing phase when software is tested rigidly, whenever a failure occurs it makes the software system more likely to fail or make errors. It is important to get a control over the errors so that the execution of a software process over time remains in-control state. To identify and remove the variations in software development process and also to monitor the software reliability various Statistical Process Control (SPC) tools are used. Statistical Process Control comprises of seven tools in which, Control charts are, perhaps, most technically sophisticated. The basic idea behind using any control chart is to identify and monitor a process from any unusual causes of variations.

Various control charts are available for monitoring a process in which Shewhart control charts are the most easy to use amongst the control charts. Basically, two types of Shewhart control charts were developed to analyze the process variation i.e. control charts for variables data and control charts for attribute data. Shewhart charts for variable data, e.g. $\overline{X}$ and R charts are powerful tools for monitoring a process but can only be used to monitor those quality characteristics that can be measured or expressed on a numerical scale. However, some quality characteristics can be observed only as attributes that confirms to the requirement or they do not confirm. The Shewhart charts for attribute can be used to monitor discrete measurement that can be modeled by the binomial or the Poisson distribution. The four commonly used attribute charts for this purpose are $p-chart$, $np-chart$, $u-chart$ and $c-chart$ [4]. These charts are based on the 3-sigma limits and normal approximation. Since the conventional Shewhart control charts for attribute data suffer from the limitation when the nonconformities rate of occurrence is very small. Due to this most of the time the lower control limit has to be fixed at zero. Many researchers like Xie and Goh [17], Wetherill and Brown [16], and Montgomery [12] suggested the use of exact probability limits in place of the usual three-sigma limits while developing the upper, lower and central control limits. Jalote and Saxena [7] also suggested the use of tighter control limits for software process. Using the exact probability limits each point has an equal chance of falling above or below the upper and lower control limits respectively. Over the years, SPC has been widely used in manufacturing industries for the purpose of improving and controlling processes and a little research is available on their use to monitor the software failure process.

For monitoring the software process, Cumulative Quantity Control (CQC) chart is used which was proposed by Chan et al. [3] and is based on the concept of exact probability limit. This procedure is based on monitoring the cumulative number of quantity between the observations of two nonconforming events in a manufacturing process. Since the quantity produced between two nonconforming observations is related to the time between failures in software reliability study, this approach can be adopted for monitoring software process. As a brief record of related research, various authors have suggested the use of control charts for reliability monitoring. Xie et al. [18] used t_r charts to monitor the failure process of systems to detect any change in the intensity function for the exponentially distributed inter-failure times. Further it gave a clear indication that after detecting a certain number of failures the process average has been improved and can be easily investigated through the control chart. Hsu and Shu [6] showed the use of two stage procedure for controlling exponential failure process. Prasad et al. [14] used a control chart mechanism for monitoring the number of faults detected using the mean value function of Goel-Okumoto model. The non-linear estimation procedure used by Prasad et al. [14] for solving the mean value function of Goel-Okumoto type for the inter-failure time data is based on manually solving the likelihood function and estimating the parameters numerically using the iterative Newton-Raphson approach. The failure control chart only indicates about the early detection of software failure compared with

the Xie et al. [18]. In their study, Prasad et al. [14] acknowledge the existence of out of control points, but do not explain about the declination of reliability at those points. In this study, we incorporate the concept of reliability trend which allows us to easily differentiate between the controlled and out of control faults during the testing, further the concept of testing effort has been utilized which allow the removal of faults at those out of control points need to be done for a better reliability growth. Here, unlike previous propositions, we show that the estimation procedure for inter-failure time can be performed using the available statistical packages in the market.

When a software system is in normal operation, occurrences of failure are random which are caused by, for example, poor coding practices, human factor, and lack of skilled testing. The failures of software system at certain point of time make the system buggy which results in a decrease of reliability. Various trend tests can be performed for checking the reliability in the grouped data or time-series which can be used to determining whether the software system undergoes reliability growth or decrease through a graphical evaluation of the data. Software reliability growth model is one of the fundamental mathematical approaches to evaluate the performance of software. To predict the content of faults in software various software reliability growth models (SRGMs) have been proposed. Some of them depict exponential growth curve while other follows S-shaped growth curve depending on the nature of growth phenomenon during the testing process. In the last decades many authors developed software reliability growth models (SRGMs) based on the non-homogeneous process [2, 5, 8, 9, 10, 19]. SRGMs are useful for evaluating how software reliability improves as faults are detected and removed. It can be used to estimate when a particular level of reliability is expected to attain and also helps in determining when to stop the testing process [9, 10]. In this paper we have studied the behavior of expected number of software failures when the underlying mean value function is of Goel-Okumoto type and formulated a control chart showing the behavior of faults detected during the testing whether they are in controlled state or not, further the reliability level at those points has been assessed using various trend tests.

The rest of the paper has been structured as follow, Section 2 describe the NHPP based SRGM and Goel-Okumoto model. A detailed analysis of the software failure data from different projects is presented in Section 3. In Section 4 the calculation of control limits is presented and various trend tests are performed from which the reliability of software can be assessed. In Section 5, conclusion is given followed by acknowledgement and references in Section 6 and 7 respectively.

2. NHPP BASED SOFTWARE RELIABILITY GROWTH MODELS

The Non-Homogeneous Poisson Process (NHPP) based on software reliability growth models (SRGMs) have been widely used in analyzing and estimating the framework for describing the software failure phenomenon during testing. Several mathematical models have been developed in the literature to monitor the fault removal process and to predict the reliability of the software. NHPP based SRGMs are basically classified into two categories: continuous and discrete. Among the discrete process, counting process is used to describe the occurrence of an event means the failure of the system in a given time. In reliability engineering a Poisson process is widely used to describe a counting process. As a general class of stochastic models, NHPP has been used for the studying hardware and software reliability problem successfully. These NHPP-based SRGMs are useful to describe certain trends in the failure processes such as growth and deterioration of reliability. Therefore, an application of NHPP models for the analysis of software process is easily implemented [1, 9, 10].

A counting process $(N(t), t \geq 0)$ representing the cumulative number of failure occurrence experienced up to time t, i.e., $N(t)$ is said to be an NHPP with intensity function $\lambda(t)$ if it satisfies the following conditions:

- There are no failures experienced at $t = 0$, i.e. $N(0) = 0$.
- The counting process has independent increments, that is, the number of failure experienced in $(t, t + \Delta t]$, i.e., $N(t + \Delta t) - N(t)$, is independent of history.
- The probability that a failure will occur during $(t, t + \Delta t]$ is $\lambda(t)\Delta t + o(\Delta t)$, i.e. $\Pr[N(t + \Delta t) - N(t) = 1] = \lambda(t) + o(\Delta t)$.

The function $o(\Delta t)$ is defined as $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$.

- The probability that more than one failure will occur during $(t, t + \Delta t]$ is $o(\Delta t)$, that is, $\Pr[N(t + \Delta t) - N(t) > 1] = o(\Delta t)$.

Based on the above assumptions, it can be shown that the probability that $N(t)$ is an integer k and expressed by:

$$\Pr[N(t) = k] = \frac{[m(t)]^k e^{-m(t)}}{k!}, k \geq 0$$

where $m(t)$ is called mean value function and represents cumulative number of failure in $(0, t]$. By the definition of mean value function, $m(t)$, can be expressed in terms of the failure intensity of the software, i.e., $m(t) = \int_0^t \lambda(s) ds$. Inversely knowing $m(t)$, the failure intensity can be obtained as, $\lambda(t) = \frac{dm(t)}{dt}$ and the reliability of the software in the time interval of length x is given as:

$$R(x \mid t) = e^{-(m(t+x) - m(t))}.$$

2.1 Model Description: Goel-Okumoto Model

One simple class of calendar time model is the Goel-Okumoto model [5], which has an exponential growth curve for the cumulative number of failures experienced. This model is based on the assumption that the failure occurrence is modeled by NHPP and the failure intensity is proportional to the remaining number of faults in the software describing an exponential decay rate function.
It can be described mathematically with the following differential equation:

$$\frac{dm(t)}{dt} = b(a - m(t)) \qquad (1)$$

Solving above with initial condition $m(0) = 0$, we get the expected number of faults at time t as

$$m(t) = a(1 - e^{-bt}) \;\; a > 0, 0 < b < 1 \qquad (2)$$

where a is the expected total number of faults to be eventually detected and b represents the fault detection rate [9, 10]. In the next section the parameter estimation has been performed.

3. PARAMETER ESTIMATION

The procedure examined here deals with the estimation of the parameters of Goel-Okumoto model when the data is about the time between failure occurrences. Here we estimate the model parameters using the maximum likelihood method making use of software package SAS [15]. The first data set (DS-1) we employed was from Real-Time Command & Control system [13]. Its size was about 21,700 and there are a total of 136 numbers of failures, which includes the failure number identifying a particular failure and the failure interval containing the time elapsed from the previous failure to the current failure. Table 1 gives the complete data set. It is also noted that the phase represented by the sample 1 to 136 were obtained during the system test operations. The second data set (DS-2) was obtained from the application Real-Time Command and Control system [13]. Its size was about 27,700 and there are 54 numbers of failures recorded during the system test operation phase and is given in Table 2.

Table 1. DS-1

Fault	TBF	Fault	TBF	Fault	TBF	Fault	TBF	Fault	TBF	Fault	TBF	Fault	TBF	Fault	TBF
1	3	18	120	35	227	52	21	69	529	86	860	103	108	120	22
2	30	19	26	36	65	53	233	70	379	87	983	104	0	121	75
3	113	20	114	37	176	54	134	71	44	88	707	105	3110	122	482
4	81	21	325	38	58	55	357	72	129	89	33	106	1247	123	5509
5	115	22	55	39	457	56	193	73	810	90	868	107	943	124	100
6	9	23	242	40	300	57	236	74	290	91	724	108	700	125	10
7	2	24	68	41	97	58	31	75	300	92	2323	109	875	126	1071
8	91	25	422	42	263	59	369	76	529	93	2930	110	245	127	371
9	112	26	180	43	452	60	748	77	281	94	1461	111	729	128	790
10	15	27	10	44	255	61	0	78	160	95	843	112	1897	129	6150
11	138	28	1146	45	197	62	232	79	828	96	12	113	447	130	3321
12	50	29	600	46	193	63	330	80	1011	97	261	114	386	131	1045
13	77	30	15	47	6	64	365	81	445	98	1800	115	446	132	648
14	24	31	36	48	79	65	1222	82	296	99	865	116	122	133	5485
15	108	32	4	49	816	66	543	83	1755	100	1435	117	990	134	1160
16	88	33	0	50	1351	67	10	84	1064	101	30	118	948	135	1864
17	670	34	8	51	148	68	16	85	1783	102	143	119	1082	136	4116

Table 2. DS-2

Fault	TBF	Fault	TBF	Fault	TBF	Fault	TBF	Fault	TBF	Fault	TBF
1	191	10	390	19	625	28	3551	37	661	46	300
2	222	11	275	20	912	29	800	38	50	47	9021
3	280	12	360	21	638	30	3910	39	729	48	2519
4	290	13	800	22	293	31	6900	40	900	49	6890
5	290	14	1210	23	1212	32	3300	41	180	50	3348
6	385	15	407	24	612	33	1510	42	4225	51	2750
7	570	16	50	25	675	34	195	43	15600	52	6675
8	610	17	660	26	1215	35	1956	44	0	53	6945
9	365	18	1507	27	2715	36	135	45	0	54	7899

Table 3. Parameter Estimates

Parameters	DS-1	DS-2
a	142.863213	55.93
b	3.4218E-05	3.096E-05

Parameter estimation is important in software reliability prediction, once the analytical solution of mean value function of a given model is known, the parameter in the solution can be determined easily. Solving the equation (1) using full information maximum likelihood (FIML) method we get the estimates of the Goel-Okumoto model. Table 3 shows the estimates of parameters for DS-1 and DS-2. We can obtain various quantities of interest after substituting a and b.

4. CONTROL LIMIT AND TREND TEST EVALUATION

4.1 Control Chart

Statistical Process Control (SPC) is used to check the behavior of the mean value function of the fault detection function. It allows the software testing team to apply the appropriate testing efforts when the fault detection process is out of control. In order to create the control limits for the mean value function of Goel-Okumoto model, CQC chart is used. The chart is based on the fact that if the failure occurrence process can be described by a Poisson process with mean occurrence rate λ, and the time between failures will be exponential. The cumulative distribution function of exponential distribution with parameter λ is given by

$$F(t;\lambda) = 1 - e^{-\lambda t} \qquad (3)$$

The traditional false alarm probability for the CQC chart is normally taken as 0.27% (although any other false alarm probability can also be used). If the false alarm probability is assumed as α, then the control limits of CQC chart can be obtained as [3]:

$$UCL = T_u = -\frac{\ln(\alpha/2)}{\lambda} \quad LCL = T_l = -\frac{\ln(1-\alpha/2)}{\lambda} \quad CL = T_c = -\frac{\ln(0.5)}{\lambda} \qquad (4)$$

The control limits of CQC chart can be defined in such a manner that if the plotted point lies between the calculated control limits, it indicates that the process is in-control state. If the point lies above the UCL, it indicates that the failure occurrence rate may have decreased which resulted in better quality. If the point lies below the LCL, it indicates the failure occurrence rate may have increased i.e. an alarming signal.

For the Goel-Okumoto model the control limits can be evaluated as follow. The mean value function for the Goel-Okumoto model is given by $m(t) = a(1-e^{-bt})$. Equating the pdf of above mean value function to 0.99865, 0.5 and 0.00135 respectively and solving for t_U, t_C and t_L.

$$1 - e^{-bt_U} = \frac{\alpha}{2} = 0.99865$$

$$1 - e^{-bt_C} = 0.5$$

$$1 - e^{-bt_L} = 1 - \frac{\alpha}{2} = 0.00135 \qquad (5)$$

For each data sets DS-1 and DS-2, the control limits UCL (t_U), CL (t_C) and LCL (t_L) are shown in Table 4.

Given the n-inter failure data the value of mean value function $m(t)$ is calculated at t_U, t_C and t_L. Table 5 shows the values of mean value function at t_U, t_C and t_L for the respective data sets.

Table 4. Control Limits

Data Set	UCL (t_U)	CL (t_C)	LCL (t_L)
DS-1	193105	20256.85	39.47967
DS-2	213425.4	22388.47	43.63411

Table 5. Mean Value Control Limits

Data Set	$m(t_U)$	$m(t_C)$	$m(t_L)$
DS-1	142.6703	71.43161	0.192865
DS-2	55.85449	27.965	0.075505

In order to create control chart for detecting the alarming points where more effort is to be given while removing a fault a plot is created using the consecutive differences of the mean value function which is plotted against the faults detected during the whole testing time. Table 6 and 7 shows the predicted values of mean value function and the consecutive differences for data set DS-1 and DS-2 respectively. As per the rules for detecting the out of control points of a software testing process those values of non-cumulative value of mean value function lies below the $m(t_L)$ will trigger an alarming signal. For each data set DS-1 and DS-1 the control chart is shown in Figure 1 and 2.

Table 6. Consecutive Differences of Mean Value Function (DS-1)

Fault	$m(t)$	CD	Fault	$m(t)$	CD	Fault	$m(t)$	CD	Fault	$m(t)$	CD
1	0.014665	0.014665	35	23.79323	0.928473	69	59.68052	1.519423	103	109.2605	0.12441
2	0.161229	0.146564	36	24.05776	0.264537	70	60.75232	1.071797	104	109.2605	0
3	0.711938	0.55071	37	24.7711	0.713338	71	60.87585	0.123532	105	112.6527	3.392226
4	1.105387	0.393448	38	25.00524	0.234138	72	61.23696	0.361104	106	113.9147	1.261958
5	1.662116	0.556729	39	26.83392	1.828679	73	63.46829	2.23133	107	114.8339	0.919187
6	1.705594	0.043478	40	28.01891	1.184992	74	64.25224	0.783955	108	115.4973	0.663397
7	1.715254	0.00966	41	28.39946	0.380552	75	65.05508	0.802843	109	116.3045	0.80721
8	2.154081	0.438827	42	29.42494	1.025474	76	66.45084	1.395756	110	116.5262	0.221721
9	2.692304	0.538223	43	31.16593	1.740995	77	67.18204	0.731202	111	117.175	0.648847
10	2.76423	0.071927	44	32.13631	0.970383	78	67.59525	0.413212	112	118.7895	1.614488
11	3.424228	0.659998	45	32.8802	0.743892	79	69.69785	2.102598	113	119.155	0.365415
12	3.66259	0.238362	46	33.60415	0.723941	80	72.18568	2.487825	114	119.466	0.311083
13	4.02887	0.36628	47	33.62657	0.022429	81	73.25373	1.068053	115	119.8204	0.354358
14	4.142838	0.113968	48	33.92147	0.294891	82	73.95521	0.701481	116	119.9164	0.095994
15	4.654538	0.5117	49	36.92124	2.999777	83	77.9715	4.016293	117	120.6807	0.764322
16	5.070083	0.415545	50	41.70729	4.78605	84	80.29158	2.320077	118	121.3887	0.708023
17	8.193192	3.123109	51	42.21828	0.510984	85	83.99499	3.703407	119	122.1693	0.780528
18	8.745033	0.551842	52	42.29057	0.072295	86	85.70209	1.707098	120	122.1848	0.015572
19	8.8643	0.119267	53	43.08923	0.798655	87	87.59279	1.890707	121	122.2378	0.053
20	9.385991	0.52169	54	43.54566	0.456436	88	88.91385	1.321062	122	122.5752	0.337385
21	10.86214	1.476153	55	44.75153	1.205863	89	88.97474	0.060885	123	126.0608	3.485577
22	11.11033	0.24819	56	45.39733	0.6458	90	90.55175	1.577014	124	126.1182	0.057396
23	12.19684	1.086506	57	46.18124	0.783909	91	91.83178	1.280031	125	126.1239	0.005729
24	12.50052	0.303683	58	46.28374	0.102502	92	95.73115	3.899367	126	126.7263	0.602347
25	14.36943	1.868906	59	47.49552	1.211786	93	100.2274	4.496231	127	126.9298	0.203561
26	15.15842	0.78899	60	49.90549	2.409963	94	102.3064	2.079065	128	127.3548	0.424944
27	15.20211	0.04369	61	49.90549	0	95	103.4596	1.153176	129	130.2978	2.943075
28	20.1113	4.909185	62	50.64051	0.735029	96	103.4758	0.016176	130	131.6476	1.349756
29	22.60579	2.494494	63	51.67603	1.035513	97	103.826	0.350198	131	132.0416	0.393959
30	22.6675	0.061709	64	52.80783	1.131803	98	106.1578	2.331841	132	132.2789	0.23731
31	22.81547	0.147971	65	56.49579	3.68796	99	107.2283	1.070503	133	134.0901	1.81124
32	22.8319	0.01643	66	58.08571	1.589919	100	108.9358	1.707504	134	134.4315	0.341408
33	22.8319	0	67	58.11471	0.029004	101	108.9707	0.03481	135	134.9525	0.521
34	22.86475	0.032853	68	58.1611	0.046386	102	109.1361	0.165437	136	135.9918	1.039248

*CD: Consecutive Differences

Table 7. Consecutive Differences of Mean Value Function (DS-2)

Fault	$m(t)$	CD	Fault	$m(t)$	CD	Fault	$m(t)$	CD
1	0.329758	0.329758	19	14.23491	0.814656	37	40.05507	0.32822
2	0.710595	0.380837	20	15.39573	1.160817	38	40.07963	0.024555
3	1.187212	0.476617	21	16.18852	0.792797	39	40.43336	0.353733
4	1.676515	0.489303	22	16.5474	0.358876	40	40.8592	0.425838
5	2.161444	0.484929	23	17.99779	1.450392	41	40.94295	0.083753
6	2.79854	0.637095	24	18.70975	0.711955	42	42.78054	1.837592
7	3.727936	0.929397	25	19.4795	0.769758	43	47.81751	5.03697
8	4.704553	0.976616	26	20.82517	1.345668	44	47.81751	0
9	5.280162	0.57561	27	23.65534	2.830172	45	47.81751	0
10	5.888051	0.607889	28	27.0155	3.360153	46	47.89251	0.075
11	6.3123	0.424249	29	27.72286	0.707358	47	49.85109	1.958571
12	6.862248	0.549949	30	30.93885	3.215996	48	50.30715	0.456069
13	8.062632	1.200383	31	35.74585	4.806997	49	51.38729	1.080137
14	9.822647	1.760016	32	37.70618	1.960328	50	51.83458	0.447288
15	10.39999	0.577341	33	38.53853	0.832349	51	52.16883	0.334253
16	10.47041	0.070426	34	38.64321	0.10468	52	52.87105	0.702218
17	11.38989	0.919477	35	39.65899	1.015782	53	53.46287	0.591822
18	13.42025	2.030362	36	39.72685	0.067864	54	53.99811	0.535232

*CD: Consecutive Differences

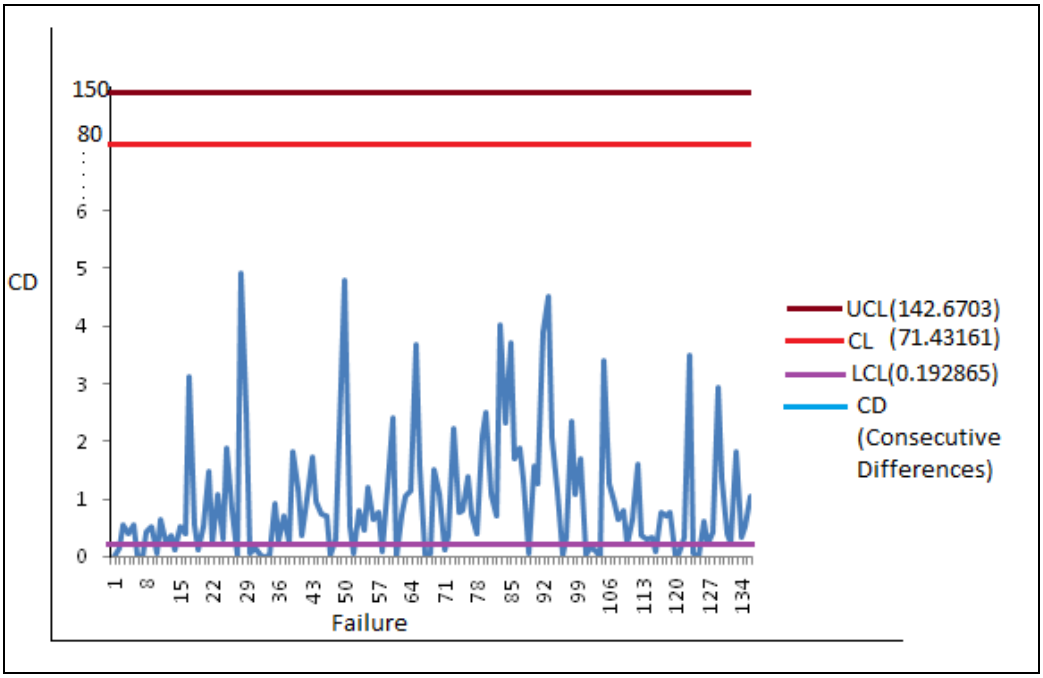

Figure 1. Control Charts for DS-1

From DS-1, as seen from Figure 1, we find that 31 point out of 136 falls outside the lower control limit; it is a clear indication that for these faults detected at various time frames, these detected faults require more effort for removal. For DS-2, as shown in Figure 2, we find 6 point out of 54 below the lower control limit; it is obvious that there are assignable causes which lead to significant process deterioration and should be carefully examined for achieving reliability. In order to validate the approach we perform various trend tests, to determine whether the software process undergoes reliability growth or decrease at the points detected out of control point. In the next section application of trend test has been done.

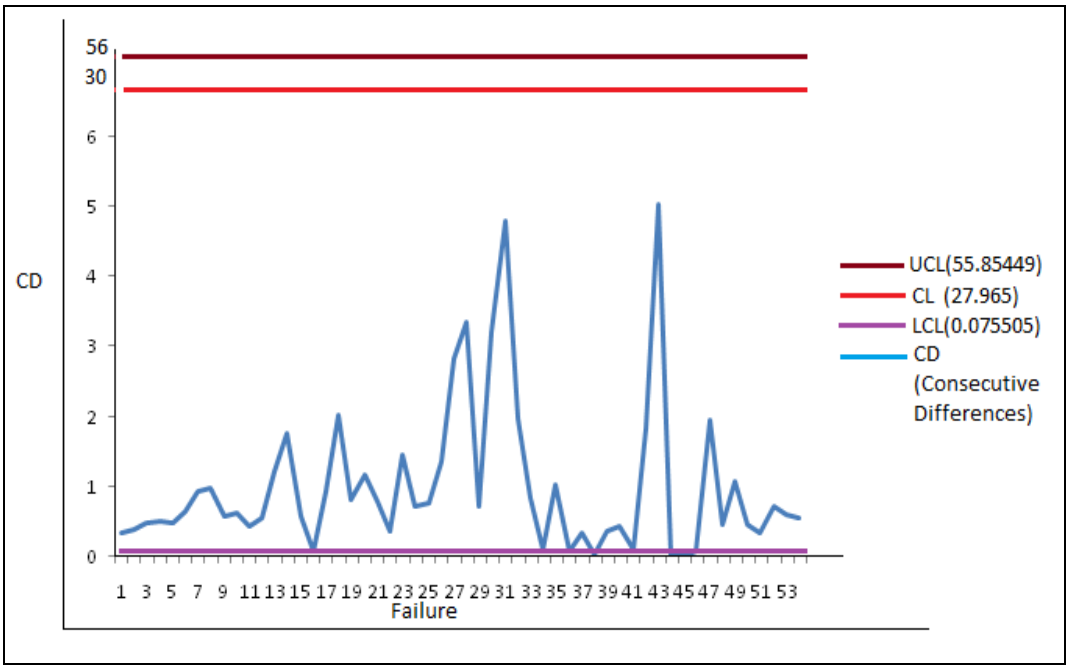

Figure 2. Control Chart for DS-2

4.2 Trend Test: Reliability Evaluation

For the inter failure times there are commonly two tests carried out: the Laplace test and the arithmetical mean test. The Laplace test and the arithmetical mean test are most commonly used trend tests which can be used to determine whether the system undergoes reliability growth or decrease, and have been further discussed in detail [11]. We will calculate the Laplace factor $u(i)$ of the observed n-inter failure times, negative values indicate decreasing failure intensity that is growth in reliability and positive values suggest a decrease in reliability. And for the arithmetical mean we will calculate $\tau(i)$ of the observed inter failure times. An increasing series of $\tau(i)$ indicates reliability growth and, conversely, a decreasing series suggests reliability decrease. The mathematical expressions for the Laplace test and the arithmetic mean test are shown in equation (6) and (7).

$$u(i) = \frac{\frac{1}{i-1}\sum_{n=1}^{i-1}\sum_{j=1}^{n}\theta_j - \frac{\sum_{j=1}^{i}\theta_j}{2}}{\sum_{j=1}^{i}\theta_j\sqrt{\frac{1}{12(i-1)}}} \qquad (6)$$

$$\tau(i) = \frac{1}{i}\sum_{j=1}^{i}\theta_j \qquad (7)$$

With reference to the data sets in Table 1-2, the Laplace trend test and the arithmetic mean test results are shown in Figures 3-4. As seen from the Figures 3-4, the Laplace factor values for the faults that are pointed out of control from the Figures 1-2 shows an increase in the Laplace factor $u(i)$ from the previous value. For example, with reference to the Figure 1 fault number 6 and 7 detected during the testing are below the lower control limit value hence are out of control, for these

detected faults the Laplace factor value goes from negative to positive that is a decrease in reliability. Similarly form the Figure 2 fault number 16 is out of control and for this fault the Laplace factor value is more than its preceding one which shows and decrease in reliability growth.

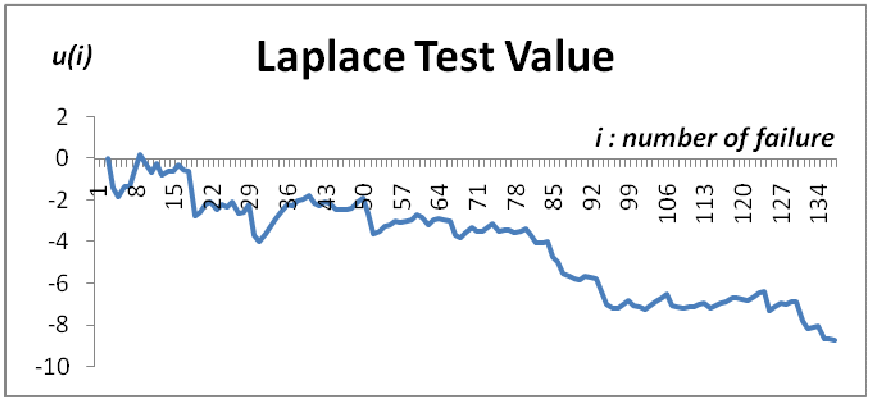

Figure 3. Laplace Test for DS-1

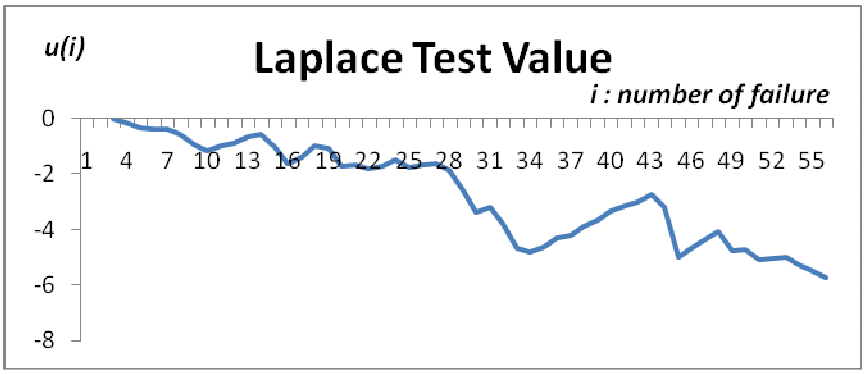

Figure 4. Laplace Test for DS-2

We can perform the same validation using the arithmetic mean test, for example as seen from the Figure 5 the arithmetic mean value shows a decrease for the fault number 6 and 7 which indicates a decrease in reliability growth and as shown in Figure 6 for fault number 16 we find a decrease in the mean value.

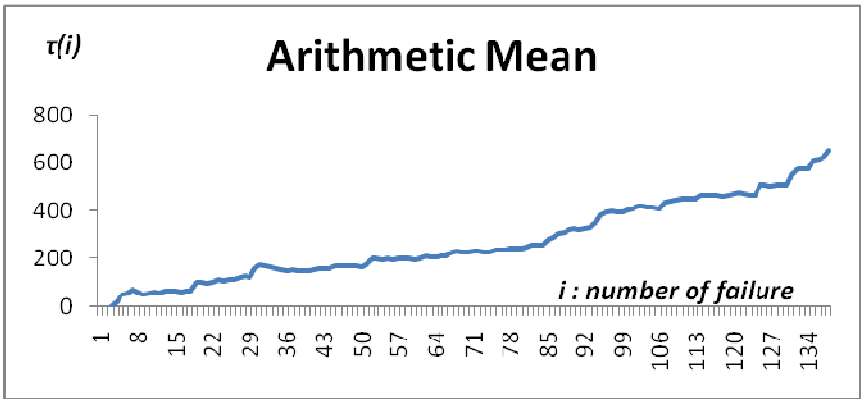

Figure 5. Arithmetic Mean Test for DS-1

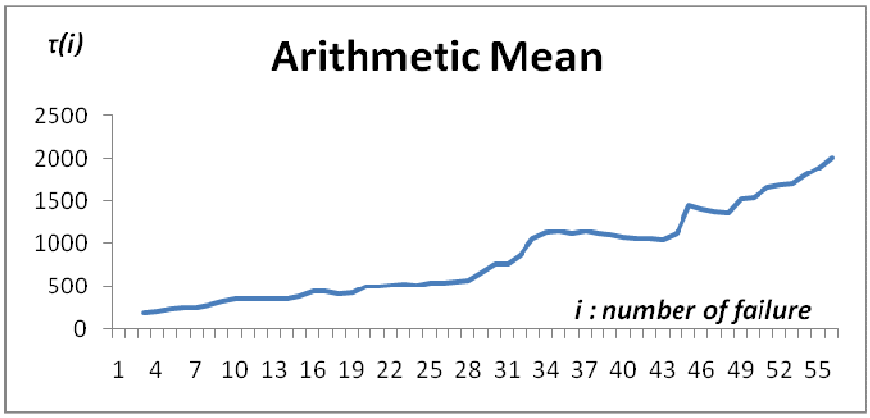

Figure 6. Arithmetic Mean Test for DS-2

The Laplace and the Arithmetic mean trend tests shows a positive behavior with respect to the mean value control chart for the failure detection function, that is, during the testing process if the detected faults lie below the lower limit of the chart, for those faults the reliability of software system will show a declination and hence these out of control faults requires more testing efforts. The behavior of the reliability for these out of control points can be easily justified using the trend tests.

5. CONCLUSION

This paper discusses the use of control charts in the software quality control process like reliability and testing. Until now, statistical control charts have been widely used in the manufacturing industry for the process monitoring tool. We assert that, for applying control charts to the software process standard charting techniques used in manufacturing processes differ significantly since it focuses on the usual 3σ-control limits. To deal with the problems associated with the conventional quality control charts that are usually used to monitor production processes a new procedure specifically for the software process can be used. In this paper, we have studied the control charting technique to monitor the failure detection process of a software system. To find the control limits of software reliability growth model (SRGM) based on NHPP an analysis has been presented. We have shown how the fault detection and control limits for the mean value function are associated and how the out of control (limit) points are determined using the CQC chart methodology. Further, the analysis also shows the determination of reliability level at the points lying outside the control limit. Laplace trend test and Arithmetic mean test have been applied for the confirmation of reliability level.

ACKNOWLEDGEMENT

The research work presented in this paper is supported by grants to the first author from University of Delhi, R&D Grant No - RC/2014/6820, Delhi, India.

REFERENCES

[1] Anand A., (2013), A Study of Innovation Adoption and Warranty Analysis in Marketing and Successive Software Releases, Ph.D. Thesis, University of Delhi, Delhi, India.

[2] Anand, A., Aggrawal. D., Das. S. and Dhiman. D., (2013), Computation of Discrepant Faults Using Flexible Software Reliability Growth Modeling Framework, CDQM, 16(2),15-27, Serbia.

[3] Chan, L. Y., Xie, M. and Goh, T. N., (2000), Cumulative quantity control charts for monitoring production processes, International Journal of Production Research, 38 (2), 397-408.
[4] Florac W.A. and Carleton A. D., (1999), Measuring the Software Process: Statistical Process Control for Software Process Improvement (SEI Series in Software Engineering), Addison-Wesley, New York.
[5] Goel A. L. and K. Okumoto, (1979), Time-dependent error-detection rate model for software and other performance measures, IEEE Transactions on Reliability, R-28(3), 206-211.
[6] Hsu, Bi-Min. and Shu, Ming-Hung., (2011), A two-phase method for controlling Erlang-failure processes with high reliability, Journal of Applied Statistics, 38(4), 717-734.
[7] Jalote, P. and Saxena, A. (2002), Optimum control limits for employing statistical process control in software process, IEEE Transactions on Software Engineering, 28 (12), 1126-1134.
[8] Kapur P. K., Garg R. B., (1992), Software Reliability Growth Model for an Error Removal Phenomenon, Software Engineering Journal, (7), 291-294.
[9] Kapur P. K., Garg R. B., Kumar S., (1999), Contribution to Hardware and Software Reliability, World Scientific Publishing Co. Pte. Ltd., Singapore.
[10] Kapur, P. K., Pham, H., Gupta, A. and Jha, P. C., (2011), Software Reliability Assessment with OR Applications. Springer Series in Reliability Engineering, Springer-Verlag London.
[11] Lyu, M. R., (1996), Handbook of Software Reliability Engineering. New York: McGraw Hill.
[12] Montgomery, D. C., (2001), Introduction to Statistical Quality Control. John Wiley and Sons Inc., New York.
[13] Musa, John D., (1980), Software Reliability Data, Cyber Security and Information Systems Information Analysis Center.
[14] Prasad S. R., Rao B. S., Kantham R. R. L., (2011). Assessing software reliability using inter-failures time data, International Journal of Computer Applications, 18(7).
[15] SAS Institute Inc. (2004) SAS/ETS User's Guide Version 9.1, SAS Institute Inc., Cary, NC.
[16] Wetherill, G. B. and Brown, D. W., (1991), Statistical process Control-Theory and Practice, Chapman and Hall, London.
[17] Xie, M. and Goh, T. N. (1993), Improvement detection by control charts for high yield processes, International Journal of Quality and Reliability Management, 10 (7), 23-29.
[18] Xie, M., Goh. T. N., Ranjan. P., (2002), Some effective control chart procedures for reliability monitoring, Reliability engineering and System Safety, 77, 143 -150.
[19] Yamada S., Ohba, M., Osaki S., (1983), S-shaped Software Reliability Growth Modeling for Software Error Detection, IEEE Transaction Reliability, R-32(5), 475-484.