SOFTWARE RELIABILITY USING SPC: GOMPERTZ

R.Satya Prasad , V. Surya Narayana, G. Krishna Mohan
Dept. of CS&E., ANU, Ramachandra College of Engg., KL University

Abstract

Software reliability models are very important for prediction and estimation of software reliability. In this paper the application of Gompertz Software Reliability Model with Statistical Process Control for software reliability is presented. Maximum Likelihood Estimation is used to estimate the parameters of the model. The model is applied on the cumulative quantities of Time domain software failure data collected from different sources.

Keywords: Statistical Process Control, Maximum Likelihood Estimation, Gompertz, Software Reliability Growth Model, Non Homogenous Poisson Process.

I. INTRODUCTION

Over the past four decades, the deployment of computer system has grown more intensely. Software is everywhere, but we need reliable software.Software Reliability is defined as the probability of failure-free operation of a computer program in a specified environment for a specified time.

In general SRGMs are applicable to the late stages of testing in software development. They can provide very useful information about how to improve the reliability of software products. The problem with usingSRGM to estimate failure content is that they have underlying assumptions that are often violated in practice,but empirical evidence has shown that many are quite robust despitethese assumption violations. The problem is that, because of assumption violations, it is often difficultto know which models to apply in practice.Reliability quantities have usually been defined with respect to time, although it is possible to define them with respect to other variables. Most software reliability models are formulated in terms of random processes. It is traditional and important to use Non Homogenous Poisson Process (NHPP) to model the failure process. Models considered the probability of failure times or failures experienced and the nature of the variation of the random process with time. So, most models are time-based. Specification of a model generally includes specification of a function of time such as the mean value or failure intensity function.

Both static and dynamic software reliability models exist to assess the quality aspect of software. A static model uses software metrics, like complexity metrics,results of inspections, etc. to estimate the number of defects (or faults) in the software. A dynamic modeluses the past failure discovery rate during software execution or cumulative failure profile over time toestimate the number of failures. It includes a time component, typically time between failures.

A failure is a departure from how softwareshould behave during operation according to the requirements. Failures are dynamic: The software mustbe executing for a failure to occur. A fault is a defect in a program, that when executed causes failure(s).While a fault is a property of the program, a failure is a property of the program's execution.

Dynamic models measure and model the failure process itself. Because of this, they include a timecomponent, which is typically based on recording times t_i of successive failure i and i-1.

Time may berecorded as execution time or calendar time. These models focus on the failure history of software. Failurehistory is influenced by a number of factors, including the environment within which the software is executedand how it is executed. A general assumption of these models is that software must be executed accordingto its operational profile; that is, test inputs are selected according to their probabilities of occurringduring actual operation of the software in a given environment.

The expected value function for cumulative failures can be put into two shape classes: concave andS-shaped. S-shaped models are first convex, then concave. The S-shaped growth curves start at some fixed point and increase their growth rate monotonically to reach an inflection point. After this point, the growth rate approaches a final value asymptotically. The S-shaped models reflect an assumption thatearly testing is not as efficient as later testing, so there is a period during which the failure-detectionrate increases. This period terminates, resulting in an inflection point in the S-shaped curve, when thefailure-detection rate starts to decrease.

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. The control chart is one of the seven tools for quality control. 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 only chance causes of variation. On the other hand, when assignable causes are present, then we say that the process is statistically “out-of-control.”

Control charts should be capable to create an alarm when a shift inthe level of one or more parameters of the underlying distributionor a non-random behavior occurs. Normally, such a situation willbe reflected in the control chart by points plotted outside thecontrol limits or by the presence of specific patterns. The mostcommon non-random patterns are cycles, trends, mixtures andstratification (Koutras, 2007). For a process to be in control the control chartshould not have any trend or nonrandom pattern.

II. NHPP MODELS

The NHPP group of models provides an analytical framework for describing the software failure phenomenon during testing. They are proved to be quite successful in practical software reliability engineering. They have been built upon various assumptions. If 't' is a continuous random variable with probability density function: $f(t, \theta_1, \theta_2, \ldots, \theta_k)$, and cumulative distribution function: $F(t)$.where $\theta_1, \theta_2, \ldots, \theta_k$ are k unknown constant parameters. The mathematical relationship between the pdf and cdf is given as: $f(t) = F'(t)$.

Let $N(t)$ be the cumulative number of software failures by time 't'. A non-negative integer-valued stochastic process $N(t)$ is called a counting process, if $N(t)$ represents the total number of occurrences of an event in the time interval [0, t] and satisfies these two properties:

1. If $t_1 < t_2$, then $N(t_1) \le N(t_2)$
2. If $t_1 < t_2$, then $N(t_2) - N(t_1)$ is the number of occurrences of the event in the interval $[t_1, t_2]$.

One of the most important counting processes is the Poisson process. A counting process, $N(t)$, is said to be a Poisson process with intensity λ if

1. The initial condition is N(0) = 0
2. The failure process, N(t), has independent increments
3. The number of failures in any time interval of length s has a Poisson distribution with mean λs, that is, $P\{N(t+s) - N(t) = n\} = \frac{e^{-\lambda s}(\lambda s)^n}{n!}$

Describing uncertainty about an infinite collection of random variables one for each value of 't' is called a stochastic counting process denoted by $[N(t), t \ge 0]$. The process $\{N(t), t \ge 0\}$ is assumed to follow a Poisson distribution with characteristic Mean Value Function $m(t)$, representing the expected number of software failures by time 't'. Different models can be obtained by using different non decreasing $m(t)$. The derivative of $m(t)$ is called the failure intensity function $\lambda(t)$.

A Poisson process model for describing about the number of software failures in a given time (0, t) is given by the probability equation.

$$P\left[N(t)=y\right]=\frac{e^{-m(t)}[m(t)]^{y}}{y!}, \quad y=0,1,2,...$$

Where, $m(t)$ is a finite valued non negative and non decreasing function of $'t'$ called the mean value function. Such a probability model for $N(t)$ is said to be an NHPP model. The mean value function $m(t)$ is the characteristic of the NHPP model.

The NHPP models are further classified into Finite and Infinite failure models. 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: $m(t)=aF(t)$. The failure intensity function $\lambda(t)$ is given by: $\lambda(t)=aF'(t)$.

III. GOMPERTZ SOFTWARE RELIABILITY MODEL

The simplest form of a software reliabilitygrowth model is an exponential one.However, S-shaped software reliability is moreoften observed than the exponential one. Some models use a non-homogeneous Poisson process (NHPP) to model the failure process. The NHPP is characterized by its expected value function, m(t). This is the cumulative number of failures expected to occur after the software has executed for time t.Gompertz SRGM is based on an NHPP. Infact, many Japanese computer manufacturersand software houses have applied theGompertz curve model, which is one of thesimplest S-shaped software reliability growthmodels(Kececioglu, 1991). The Gompertz curve model gavegood approximation to cumulative number ofsoftware faults observed (Satoh, 2000). It takes the number of faults per unit of time as independent Poisson random variables. The Gompertzmodel equation for software reliability is,

$$m(t)=ab^{c^{t}}$$

Where, 'a' is the upper limit approached thereliability, R at time t. 0<b<1, 0<c<1 areparameters to be estimated form any one of theparameter estimation methods.

where

a is the expected total number of failures that would occur if testing was infinite.

b is the rate at which the failures detection rate decreases.

c models the growth pattern (small values model rapid early reliability growth, and large valuesmodel slow reliability growth).

The Gompertz distribution plays an important role in modeling survival times, humanmortality and actuarial tables. According to the literature, the Gompertz distribution was formulated by Gompertz (1825) to fit mortality tables. Recently, many authors have contributed to the statistical methodology and characterization of this distribution. For example, Read (1983),Gordon (1990), Makany (1991), Franses (1994) and Wu & Lee(1999).Garg*et al*. (1970) studied the properties of the Gompertz distribution and obtained themaximum likelihood estimates for the parameters.There are several forms for the Gompertz distributiongiven in the literature. Some of these are given in Johnson *et al.* (1994).Gompertz software reliability model is a popular model toestimate remaining failures. It has been widely used to estimate software error content, it is a modified model ofMoranda reliability model.

IV. MAXIMUM LIKELIHOOD ESTIMATION OF THE MODEL PARAMETERS

There are twomethods of parameter determination.Parameter prediction and parameterestimation. Parameter prediction tries toestablish the parameters of a model from theproperties of the software product and thedevelopment process. Parameter estimation isused in subsystem or system test or operationalphase where failure data are available. It is astatistical method trying to estimate modelparameters based on failure times. A number of procedures can be used to estimate the parameters of Gompertz reliability model. Among these methods, The maximum likelihood estimation has been frequently considered to estimate the parameters of the Gompertz model.

The likelihood function of the sample is given by

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

$$L = e^{-ab^{c^{t_n}}} \prod_{i=1}^{n} \left[ab^{c^{t_i}} c^{t_i} \log b \log c \right] \qquad 1$$

It is usually easier to maximize the natural logarithm of the likelihood function rather than the likelihood function itself. So, the natural logarithm of the likelihood function can be written as:

$$\log L = \sum_{i=1}^{n}\left[\log a + \log\left(b^{c^{t_i}}\right) + \log\left(c^{t_i}\right) + \log(\log b) + \log(\log c)\right] - ab^{c^{t_n}} \qquad 2$$

The first derivatives of the natural logarithm of the total likelihood function in (2) with respectto a, band c are given by:

$$\frac{\partial \log L}{\partial a} = \frac{n}{a} - ab^{c^{t_n}} \qquad 3$$

$$\frac{\partial \log L}{\partial b} = nc^{t_n} - \sum_{i=1}^{n} c^{t_i} - \frac{n}{l\,og\,b} \qquad 4$$

$$\frac{\partial \log L}{\partial c} = \frac{1}{c}\left(\frac{n}{\log c} + \sum_{i=1}^{n} t_i + \sum_{i=1}^{n} t_i c^{t_i}\log b\right) - n\log b t_n c^{t_n - 1} \qquad 5$$

By equating equation (3) and (4) to zero, the maximum likelihood estimate of a and b can be given by thefollowing estimation equation:

$$a = \frac{n}{b^{c^{t_n}}} \qquad 6$$

$$b = e^{\frac{n}{nc^{t_n} - \sum_{i=1}^{n} c^{t_i}}} \qquad 7$$

Substituting ‘b’ in the equation (5), we get

$$\frac{\partial \log L}{\partial c} = \frac{1}{c}\left(\frac{n}{\log c} + \sum_{i=1}^{n} t_i + \sum_{i=1}^{n} t_i c^{t_i}\frac{n}{nc^{t_n} - \sum_{i=1}^{n} c^{t_i}}\right) - \frac{n^2}{nc^{t_n} - \sum_{i=1}^{n} c^{t_i}} t_n c^{t_n - 1} \qquad 8$$

Obviously, it is very difficult to obtain a closed-form solution, iterative procedures must be used to solve these equations, numerically. The Newton-Raphson method is used to obtain the MLE of ‘c’. Therefore, take the 2nd derivative with respect to ‘c’ and equating it to Zero.

$$\frac{\partial^2 \log L}{\partial c^2} = \frac{1}{c}\left(\sum_{i=1}^{n} t_i^{\,2} c^{t_i - 1} \frac{n}{nc^{t_n} - \sum_{i=1}^{n} c^{t_i}} + \sum_{i=1}^{n} t_i c^{t_i} \frac{-n}{\left(nc^{t_n} - \sum_{i=1}^{n} c^{t_i}\right)^2}\left(nt_n c^{t_n - 1} - \sum_{i=1}^{n} t_i c^{t_i - 1}\right) - \frac{n}{c(\log c)^2} \right)$$

$$-\frac{1}{c^2}\left(\sum_{i=1}^{n} t_i c^{t_i} \frac{n}{nc^{t_n} - \sum_{i=1}^{n} c^{t_i}} + \sum_{i=1}^{n} t_i + \frac{n}{\log c} \right)$$

$$-n^2 t_n \frac{\left[\left(nc^{t_n} - \sum_{i=1}^{n} c^{t_i}\right)(t_n - 1)c^{t_n - 2} - c^{t_n - 1}\left(nt_n c^{t_n - 1} - \sum_{i=1}^{n} t_i c^{t_i - 1}\right)\right]}{\left(nc^{t_n} - \sum_{i=1}^{n} c^{t_i}\right)^2}$$

Thus, once the value of c ˆ is determined, an estimate of bˆis easily obtained from (7). They are then substituted in equation (6) to get an estimate of aˆ.

V. TIME DOMAIN FAILURE DATA

The success of applying and using a software reliability model depends highly on the quality and accuracy of failure data collection which in turn depends on careful planning and organization. The data which is considered has been transformed so as to suit to the model under consideration.

The following tables shows the number of errors and the inter failure time.

Table 1.Data Set #1: Data collected from (Xie*et al*., 2002).

Failure Number	Cumulative Time Between Failure(h)	Failure Number	CumulativeTime Between Failure(h)	Failure Number	CumulativeTime Between Failure(h)
1	0.3002	11	1.1534	21	2.5681
2	0.3146	12	1.2157	22	2.7388
3	0.5393	13	1.2496	23	2.7787
4	0.5529	14	1.3407	24	4.5393
5	0.5872	15	1.3625	25	5.35
6	0.7192	16	1.5178	26	5.3727
7	0.7707	17	1.775	27	5.529
8	0.809	18	1.8029	28	6.7368
9	1.019	19	1.8221	29	7.0449
10	1.1487	20	1.8634	30	7.3868

Table 2.Data Set #2: On-Line Data Entry IBM Software Package

The data reported by Ohba (1984a) are recorded from testing an on-line data entry software package developed at IBM.

Failure Number	Cumulative Inter Failure Time	Failure Number	CumulativeInter Failure Time	Failure Number	CumulativeInter Failure Time
1	0.1	6	0.7	11	1.69
2	0.19	7	0.88	12	1.99
3	0.32	8	1.03	13	2.31
4	0.43	9	1.25	14	2.56
5	0.58	10	1.5	15	2.96

VI. RESULTS

The control limits for the chart are defined in such amanner that the process is considered to be out of control whenthe time to observe exactly one failure is less than LCL or greaterthan UCL. Our aim is to monitor the failure process and detectany change of the intensity parameter. When the process isnormal, there is a chance for this to happen and it is commonlyknown as false alarm. The traditional false alarm probability is toset to be 0.27% although any other false alarm probability can beused. Assuming an acceptable probability of false alarm, the control limits can be obtained as (Xie *et al*, 2002):

$$T_U = b^{c^t} = 0.99865$$

$$T_C = b^{c^t} = 0.5$$

$$T_L = b^{c^t} = 0.00135$$

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 failure control 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', 'b' and 'c' and their control limits for the transformed data sets are as follows.

Table 3. Parameter estimates of Time domain data.

Data Set	No. of samples	Estimated Parameters

		a	B	C
XIE	**30**	30.526286	0.055202	0.500320
IBM	**15**	16.419633	0.045773	0.303497

Table 4. Control limits of Time domain data.

Data Set	No. of samples	Control Limits		
		UCL	**CL**	**LCL**
XIE	**30**	30.485076	15.263143	0.041210
IBM	**15**	16.397466	8.209817	0.022167

Table 5.Successive differences of Mean values, Xie.

Failure number	Cumulative Time between errors(days)	m(t)	successive differences
1	0.3002	19.756979	0.408055
2	0.3146	19.348924	5.378005
3	0.5393	13.970919	0.272678
4	0.5529	13.698241	0.664307
5	0.5872	13.033934	2.269495
6	0.7192	10.764439	0.774197
7	0.7707	9.990242	0.539432
8	0.809	9.450809	2.479896
9	1.019	6.970914	1.194567
10	1.1487	5.776347	0.039213
11	1.1534	5.737134	0.495318
12	1.2157	5.241816	0.251314
13	1.2496	4.990503	0.617259
14	1.3407	4.373243	0.136012
15	1.3625	4.237231	0.853994
16	1.5178	3.383237	1.052763
17	1.775	2.330474	0.092354
18	1.8029	2.238120	0.061421
19	1.8221	2.176699	0.126466
20	1.8634	2.050232	1.311908
21	2.5681	0.738324	0.161818

22	2.7388	0.576507	0.032392
23	2.7787	0.544114	0.501699
24	4.5393	0.042415	0.029316
25	5.35	0.013099	0.000424
26	5.3727	0.012675	0.002569
27	5.529	0.010106	0.008351
28	6.7368	0.001755	0.000632
29	7.0449	0.001123	0.000439
30	7.3868	0.000684	

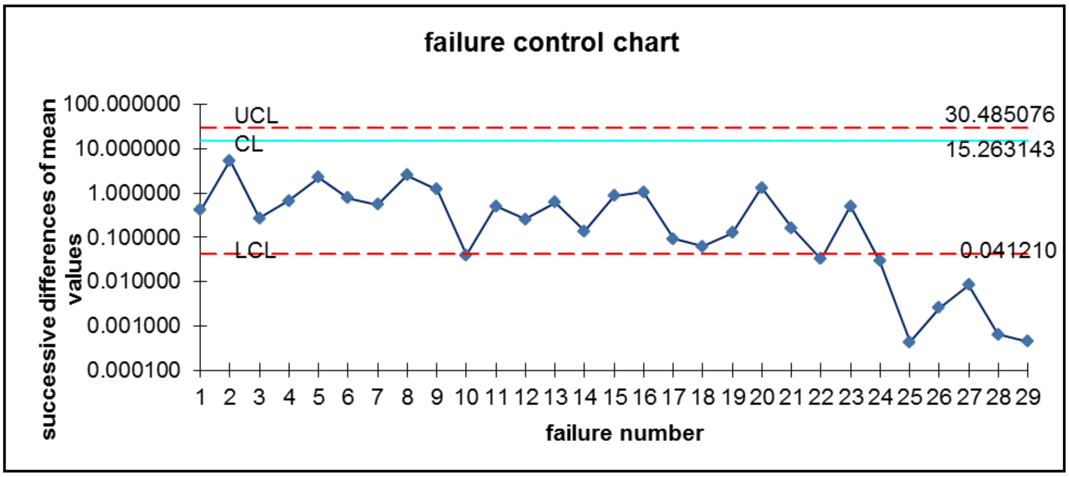

Figure 1. Failure control chart for Xie data

Table 6.Successive differences of Mean values, IBM.

Failure number	cumulative failure time	m(t)	successive differences
1	0.1	14.952484	1.208006
2	0.19	13.744478	1.574687
3	0.32	12.169791	1.190659
4	0.43	10.979132	1.438155
5	0.58	9.540977	1.013654
6	0.7	8.527323	1.322181
7	0.88	7.205142	0.943801
8	1.03	6.261341	1.165251
9	1.25	5.096091	1.063242
10	1.5	4.032848	0.657048
11	1.69	3.375800	0.826468
12	1.99	2.549332	0.659836
13	2.31	1.889496	0.394222

14	2.56	1.495274	0.466973
15	2.96	1.028301	

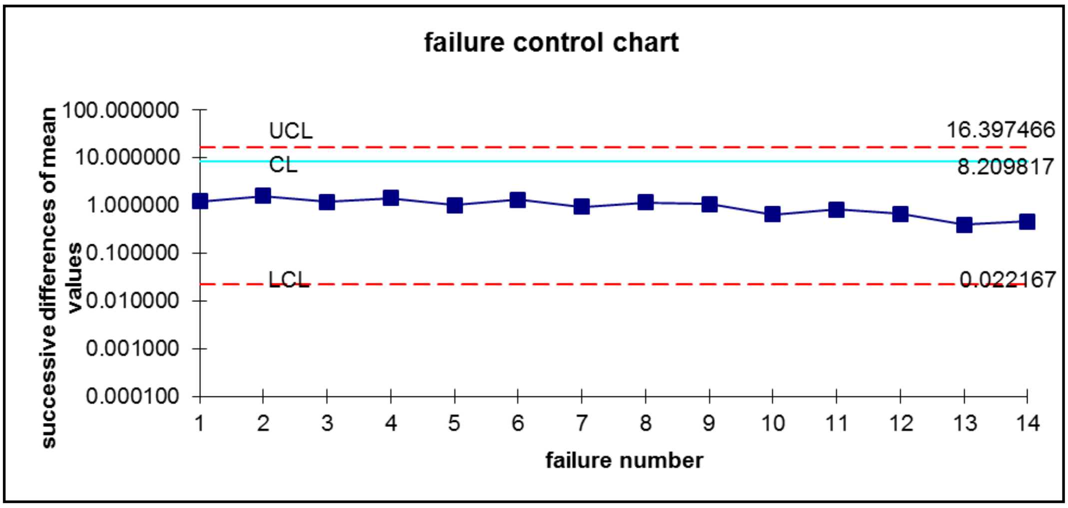

Figure 2. Failure control chart for IBM data

VII. CONCLUSION

In this paper, a theoretical review of the Gompertz SRM is provided; several mathematicalformulas of the model's characteristics are obtained. The MLE method is used to estimate the parameters of SRM. This procedure is applied on different sets of failure data collected from literature.For the Xie software failure data the first failure is observed at the 10^{th} instance of time and for the IBM data the successive differences of mean values are within the control limits. Having the successive differences within the limits indicate the failure free operation of the software under consideration. From the obtained results, we can concludethat the proposed method of using SPC techniques for assessing the quality of the software is applicable in several instances.

REFERENCES

Read, C.B. (1983). "Gompertz Distribution", Encyclopedia of Statistical Sciences, Wiley, New York.

Gordon, N.H. (1990). "Maximum Likelihood Estimation for Mixtures of two Gompertz Distributions when Censoring Occurs", Communication in Statistics – Simulation & Computation, 19, 733-747.

Makany, R. (1991), "A Theoretical Basis of Gompertz's Curve", Biometrical Journal, 33, 121-128.

Franses, P.H. (1994). "Fitting a Gompertz Curve", Journal of the Operational Research Society, 45, 109-113.

Wu, J.W. and Lee, W.C. (1999). "Characterization of the Mixtures of Gompertz Distributions by Conditional Expectation of Order Statistics", Biometrical Journal, 41, 371-381.

Garg, M.L., Rao, B.R. and Redmond, C.K. (1970). "Maximum Likelihood Estimation of the Parameters of the Gompertz Survival Function", Journal of the Royal Statistical Society C, 19, 152-159.

Johnson, N.L., Kotz, S. and Balakrishnan, N. (1994). "Continuous Univariate Distributions", Vol. 1, 2nd ed., John Wiley and Sons, (pp. 25-26, 81-85).

Gompertz, B. (1825). "On the Nature of the Function Expressive of the Law of Human Mortality and on the New Mode of Determining the Value of Life Contingencies", Phil. Trans. R. Soc.A., 115, 513-580.

Satoh, D. (2000). "A discrete Gompertz equation and a software reliability growth model". IEICE TRANSACTIONS on Information and Systems, 83(7), 1508-1513.

Kececioglu, D. (1991). "Reliability Engineering Handbook, Vol. 2, Prentice-Hall, Englewood Cliffs, N.J., 1991.

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

Ohba, M. (1984a). "Software reliability analysis models", IBM J Research Development, Vol 28(4).

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