ISSN: 0975-766X
CODEN: IJPTFI
Available Online through ***Research Article***
www.ijptonline.com

IMPLEMENTATION OF GOMPERTZ MODEL FOR ASSESSING SOFTWARE RELIABILITY

Dr.Y.Prasanth*, M.Sri Varshitha*, CH.Phani Sainath*, K.Sasank*
Department of Computer Science and Engineering, Koneru Lakshmaiah University.
Email: cpsainath96@gmail.com

Received on: 15-02-2017 *Accepted on: 29-03-2017*

Abstract

Software reliability is the likelihood for estimating the errors in the stipulated time period in any of the software system. In our modern society, computers are used in diverse areas for various applications. We get the errors, faults and failures in the reliability models. There are many models for estimating the reliability of the software. Gompertz curve will be utilized for assessing the quantity of lingering deficiencies in checking periods of programming improvement. As Gompertz bend is defined as deterministic capacity, the curve can't be connected to evaluating programming unwavering quality which is the likelihood that product framework does not come up short in a given estimated time. Here we are proposing a dependency growth model called the Gompertz software deterministic model taking into account NHPP shapes. The suggested model will be gotten from the measurable hypothesis of amazing esteem, and has an equivalent approaching property for the deterministic Gompertz bend. We will determine an EM calculation to decide the required characteristics successfully. From various illustrations with programming disappointment information saw in genuine programming advancement ventures. We assess execution of the Gompertz programming dependency model as far as dependability evaluation and disappointment forecast.

Keywords: Software reliability, Software system, Errors, Reliability models, Gompertz curve.

1. Introduction:

Amid the most recent couple of decades, programming unwavering quality building has been assuming a focal part in quantitative evaluation in programming quality. In particular, programming unfaltering quality, which has described by the probability that item system does not miss the mark in a prefixed time period, that is a champion amongst the most fundamental measures of programming quality. Specifically, software reliability, which was characterized by the likelihood that software system does not fall short within a specified time period, it stoodout amongst the most of the critical measures in software quality.Non homogeneous Poisson process (NHPP) models have especially extended much notoriety for programming builds besides specialists to evaluate the thing steady quality, also have been used to foresee the measure of remaining needs and programming discharge time.Goel and Okumoto (1979), Yamada et al. (1983) and Goel (1985) proposed earlier and comprehended NHPP-based SRMs. Considering their NHPP-based models, different experts made NHPP-based SRMs to address the situation of programming progress outlines. We portray another NHPP-based SRM by a substitute methodology from the existing system. For example, Gompertz and Logistic curve to overview the measure of additional needs. To the best of our understanding, Sakata (1974) was the essential who utilized the Gompertz and logistic swings to foresee the measure of programming distortions saw.The key thought behind his method is non-direct fall away from the faith, i.e. to fit non-straight deterministic turns to the joined number of perceived issues to minimize the aggregate of squared blunders. In any case, the apostatize examination in light of deterministic curves is not fitting to looking over software relentless quality. Right when all is said in done, figuring programming reliability requires the criticalness of the occasion of disappointment and its evenqualityt time (dissatisfaction time).With respect to backslide examination, we can't survey the failure time, and accordingly the item trustworthiness is not portrayed. Gompertz and logistic turns to evaluate the measure of remaining inadequacies. To the best of our appreciation, Sakata (1974) was the crucial who utilized the Gompertz and logistic bends to minimize the aggregate of squared slips. Taking all things into account, get ready programming constant quality requires the significance of the occasion of thwarted expectation and its event time (dissatisfaction time). As to fall away from the faith examination, we can't gage the mix-up time and thusly the thing unfazed quality is not depicted.

2. Related Work:

Software testing is a vital procedure of software improvement to guarantee the quality of software items. For vast and complex framework, testing turns out to be more unpredictable. Although much research has propelled the systems for creating test cases with high surrender scope, testing to ensure abandon free software stays troublesome. Then again an exact software reliability expectation techinque is utilized to gauge the reliability of the software product item (RattikornHewetti et al 2006). The software reliability expectation is characterized as the estimate of, how reliable an executable software framework will be sooner or later in view of information accessible at this point ondata available now.This depends on the business importance of software reliability in particular the likelihood of failure-free execution of a software system for a predefined time in a predetermined working environment. There is a contrast in the middle of estimation also, expectation of software reliability. The estimation is an appraisal of how reliable a software systemis currently in view based on test information. Prediction is generally constrained to an undertaking period preceding system test. As such, for prediction an advancement association takes data about the system a work in progress and uses some statistical regression model to conjecture the level of reliability that will be available sooner or later in testing (Peter Lakey 2002). reliability is presumably the most critical attributes characteristic from software quality.It is personally associated with imperfections, the likelihood the software without failure for a anticipate the measure of programming deficiencies perceives.The essential thought behind his approach is non-straight descend into sin, i.e. to fit non-straight deterministic curves to the joined number of recognized blames to specified period of time.

In addition to its prevalent significance, software reliabilty has ended up being the most promptly quantifiable of the attributes of software reliability. Reliability is a much wealthier measure. It is a client or user oriented rather than developer-oriented. In this way reliability measures are more helpful than fault measures (John Musa et al 1987). As faults are evacuated, as in test stage, failure intensity tends to abatement and reliabilty increase. At the point when flaws are presented amid operation or test, as in situations when new features or design or configuration changes are being brought into the system or when faults prevail repairs amid investigating, there tends to be a step increasing in failure intensity and a step decreasing in reliability. If a system is steady, as in a project that has been discharged and there is no adjustments in code, both failure intensity and reliability tend to be constant (Dong Nguyen and Thomoson 2001). As a general rule, SRE assignments are essentially connected to both software and test engineering. SRE is only a quantitative point of view of software quality managment (Koji Ohishi et al 2005 and AmritGoel 1985).

3. Software Reliability Increement Model with Gompertz Tef and Best Release Time Determination with Improved Test Efficiency:

Software quality development models are utilized to get to the nature of the product which was created, by the programming dependability change models are used after a huge time   period to get to the method for the thing which was made. Late years two or three papers delineates undaunted quality change wonder. With the time propel the amount for bumbles acknowledgment and also for the curve will grow rapidly. A huge determined attempt will be required in testing to assemble the amount of distinguishing proof or curve to mix-up for the growth of the steadfast nature of the item. Moreover a Testing-attempt is usually depicted by qunatity of persons included; amount of investigations utilized and logbook time. Precisely when the thing is slacking by date-book time then there is a need of computerized testing mechanical gatherings to deal with slacking. Use of modernized gadgets helps to extend the testing effectiveness to a more unmistakable degree.In this paper we proposed a thing relentless quality change model which unites the Gompertz testing-exertion limit and aexamination is made offlawless discharge. Evaluation is performed on two real datasets. Parameters are evaluated and the results demonstrate that our model is a perfect fit over other.

Programming is administering this world recent years. Correspondence, business and whatever other territory where there is need of programming. Each client needs a more productive and blunder free programming. By and large programming is produced by people, so there is change that blunder might proliferate through it. Dependability is thought to be one of the essential imperative variables for programming industry. Numerous papers are introduced in this setting. Unwavering quality of programming characterized as the likelihood that the product will work before it hit with a blunder in the given contingent environment.

A few creators depicted the conduct of the product unwavering quality as far as various disappointment rates. Portraying the complete programming resting regarding scientific comparisons are called unwavering quality development model. Individuals like Goel and Okumato, Yamada and Musa proposed diverse dependability development models. Amid the product testing the disappointment rate indicates diverse trademark and can't be anticipated its conduct.

The product dependability development models depict the conduct of programming testing process. Amid the change of programming various resources were consumed. The utilization bend of testing resource over the testing period can be considered as a testing effort.

The test effort can be delineated by the work spent amid the test stage, number of CPU hours and the amount of executed test cases.In a couple papers portrays the effect oftesting exertion in the product unwavering quality development model. For the most part programming testing exertion can be portrayed by Rayleigh, Weibull, exponential and logistic bend.

Testing is directed either physically or consolidating the mechanized apparatuses. Manual testing is a period expending process; however improvement of the product time bound procedure. As the time advance increasingly assets are being expended.

Manual testing can prompts postpone in the advancement of testing. By consolidating the new mechanized testing into testing can enhance the execution by certain degree. These mechanized testing instruments work effectively, by following the more blunders yet it; builds the expense of receiving new robotized device.

4. Comparing Between Maximum Likelihood And Least Square Estimators For Gompertz Software Reliabilitymodel:

Software reliability models (SRMs) are very important for estimating and predicting software reliability in the testing/debugging phase. The contributions of this paper are as follows. First, a verifiable audit of the Gompertz SRM is given. Taking into account a few software failure data, the parameters of the Gompertz programming unwavering quality model are evaluated utilizing two estimation techniques, the traditional maximum likelihood and the least square.

The strategies for estimation are assessed utilizing the MSE and R-squared criteria. The outcomes demonstrate that the least square estimation is an appealing strategy in term of predictive execution and can be utilized when the maximum likelihood method fails to give good prediction results.

5. Proposed System:

Deterministic Curve Model:

Given S(t) a chance to be the total number of software flaws recognized up to time t(>=0). At that point the Gompertz bend is given by

$$M(t) = \omega a^{b^t}, \qquad (1)$$

Where lim t->∞ S (t) = w (>0) implies basic number of issues before testing and x,y € (0,1) are optional diagram dimensions. We can see that the Gompertz curve draws a normal Sshaped twist and its motivation of accentuation is given by t= - log (- logx)/logy which depicts a schematic representation of the Gompertz curve.

The non-straight descend into sin examination is utilized to gage model parameters, i.e. x, an and b. Satoh (2000) and Satoh and Yamada (2001) showed a discrete Gompertz turn by discretizing the differential experimental verbalizations for the Gompertz wind, and related the discrete Gompertz turn to suspect the measure of recognized programming deficiencies.

As said in Introduction, the deterministic bends and their apostatize examination have a couple of bona fide burdens. To overcome such issues, Yamada (1992) considered a NHPP model with the running with mean worth point of confinement:s

$S(t)=w(x^y–x)$ (2)

Eq. (2) permits us to change the Gompertz bend with the goal that it fulfills M (0) =0. At that point the quantity of distinguished shortcomings at t =0 gets to be 0 in the NHPP model (see Fig. 1). Then again, Eq. (2) represents the trouble of figuring the greatest probability gauges because of its solid non-linearity. An alternative way to deal with developing the NHPP model with the Gompertz curve is the usage of sporadic mean worth limit. That is, whether we consider the going with mean quality limit:

$$S(t) = \begin{cases} 0, & t=0, \\ Wx^{yz}, & t>0, \end{cases} \qquad (3)$$

by then the NHPP in like manner carries on like the Gompertz twist in the extent of t > 0. Regardless, paying little heed to the likelihood that we apply Eq. (3) to examination of software dependability, the parameter estimation issue is still remained, i.e. it is hard to choose the most convincing probability gages for the mean worth cutoff in Eq. (3) by utilizing guaranteed programming issue affirmation information.

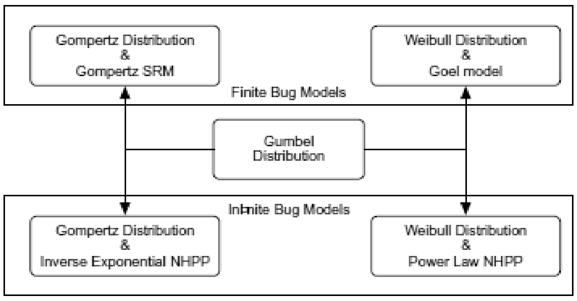

Gompertz Distribution & Gompertz SRM
Weibull Distribution & Goel model
Finite Bug Models
Gumbel Distribution
Infinite Bug Models
Gompertz Distribution & Inverse Exponential NHPP
Weibull Distribution & Power Law NHPP

Fig.5.1. Relationship among the extremal-sort NHPP models.

6. Results & Analysis:

Here, in this model we consider an excel sheet to load the required dataset and basing on the errors, information and warnings in the dataset we create a Gompertz curve and late calculating the risk assessment analysis.

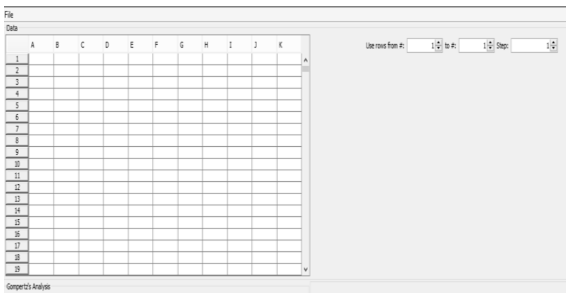


Fig: 6.1:-Creating the excel sheet.

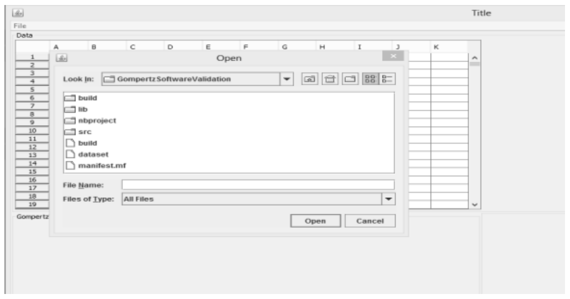
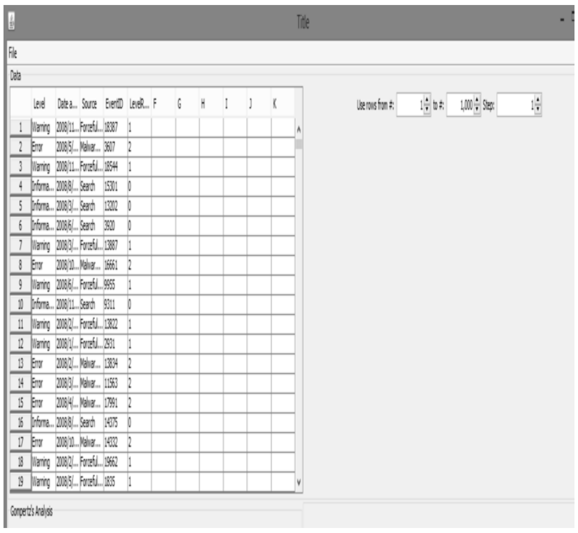


Fig: 6.2:-Loading the dataset.



Use rows from #: 1 to #: 1,000 Step: 1

	Level	Date a...	Source	EventID	LevelR...	F	G	H	I	J	K
1	Warning	2008/11...	Forceful...	18387	1						
2	Error	2008/5/...	Malwar...	3607	2						
3	Warning	2008/11...	Forceful...	18544	1						
4	Informa...	2008/8/...	Search	15301	0						
5	Informa...	2008/3/...	Search	13202	0						
6	Informa...	2008/6/...	Search	3920	0						
7	Warning	2008/3/...	Forceful...	13887	1						
8	Error	2008/10...	Malwar...	16661	2						
9	Warning	2008/6/...	Forceful...	9955	1						
10	Informa...	2008/11...	Search	9311	0						
11	Warning	2008/2/...	Forceful...	13822	1						
12	Warning	2008/1/...	Forceful...	2931	1						
13	Error	2008/2/...	Malwar...	13834	2						
14	Error	2008/3/...	Malwar...	11563	2						
15	Error	2008/4/...	Malwar...	17991	2						
16	Informa...	2008/8/...	Search	14375	0						
17	Error	2008/10...	Malwar...	14332	2						
18	Warning	2008/2/...	Forceful...	19662	1						
19	Warning	2008/5/...	Forceful...	1835	1						

Fig: 6.3:-Loaded dataset.

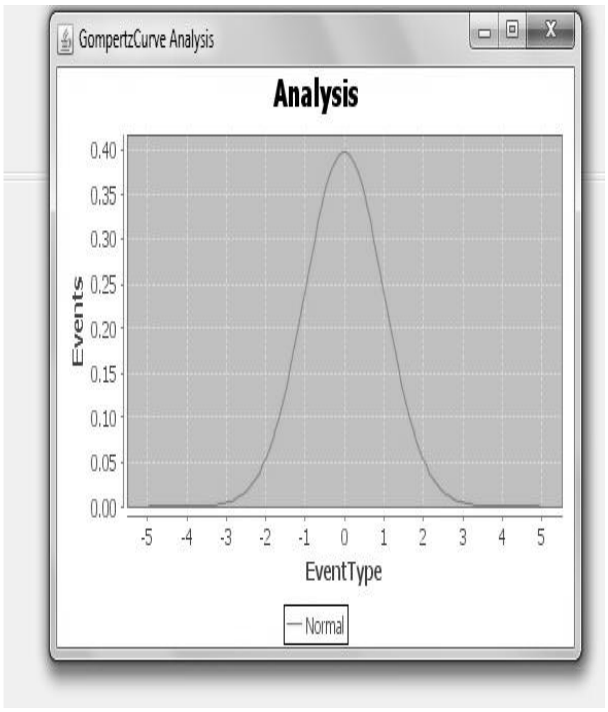
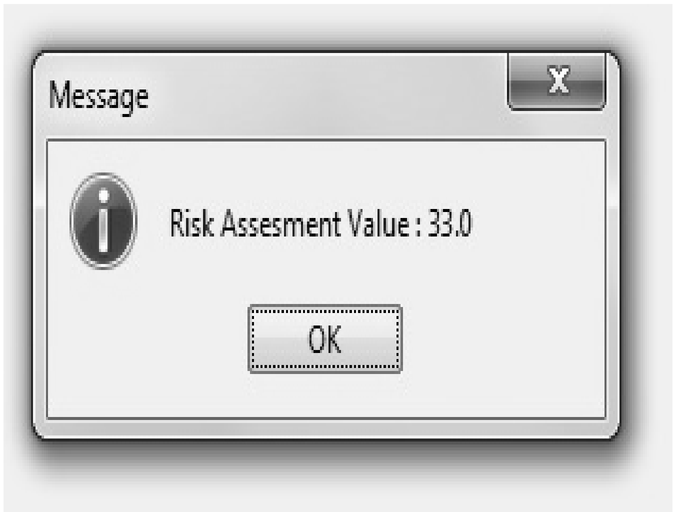


Fig: 6.4:-gompertz curve.



Risk Assesment Value : 33.0

Fig: 6.5:-Risk assessment.

7. Conclusion:

We have endeavored to propose a bound together model system of SRMs in view of the great quality hypothesis and added to the Gompertz SRM as one of the agent SRMs. As succesful parameter estimation count of calculation, we have built up the EM figuring to enroll the most awesome probability evaluations of model parameters consolidated into the Gompertz SRM.In many cases, we have checked the blending features of the EM calculation and implemented the respectability of-fit test to a genuine software shortcoming detection of the information. From certain numerical discernments, it was infrred that the proposed Gompertz SRM is to some degree drawing in separating and the current SRMs, and ought to be overseen as one of the agent NHPP-based models.We expect that the Gompertz SRM made in this paper could answer the solicitation from specialists stood up to in the item quality organization; what might we have the capacity to do with the set up example twist exhibiting approach. In future, we will break down the value and quality for the Gompertz SRM and utilization of the Gompertz in programming testing sharpen or in the product testing.

8. References:

1. Akaike, H., 1973. Information theory and an extension of the maximum likelihood principle. In: Petrov, B.N., Csaki, F. (Eds.), Proc. Second Int. Symp.on Information Theory. AkademiaiKiado, pp. 267–281.
2. Castillo, E., 1988. Extreme Value Theory in Engineering. Academic Press, San Diego.
3. Cretois, E., Gaudoin, O., 1998. New results on goodness-of-fit tests for the powerlaw process and application to software reliability. Int. J. Reliab. Qual. Safe. Eng. 5, 249–267.
4. Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Statist. Soc. B 39, 1–38.
5. Fletcher, R., 1970. A new approach to variable metric algorithms.Comput. J. 13, 317–322.
6. Galambos, J., 1987. The Asymptotic Theory of Extreme Order Statistics. Robert E. Krieger Publishing Company, Florida.
7. Gnedenko, B.V., 1943. Sur la distribution limitee du terme maximum d'uneserie aleatoire. Ann. Math. 44, 423–453.
8. Goel, A.L., 1985. Software reliability models: assumptions, limitations and applicability. IEEE Trans. Software Eng. SE-11, 1411–1423.

9. Goel, A.L., Okumoto, K., 1979. Time-dependent error-detection ratemodel for software reliability and other performance measures. IEEE Trans. Reliabl. R-28, 206–211.
10. Gokhale, S.S., Trivedi, K.S., 1998. Log-logistic software reliability growth model. In: Proc. third IEEE Int. High-Assurance Systems Eng. Sympo. (HASE-1998). IEEE CS Press, pp. 34–41.
11. Gompertz, B., 1832. On the nature of the function expressive of the law of human mortality and on a new mode of determining the value of life contingencies. Phil. Trans. Roy. Soc. Lond. 123, 513–585.
12. Gumbel, E.J., 1958. Statistics of Extremes. Columbia Press, New York. Hossain, S.A., Dahiya, R.C., 1993.Estimating the parameters of a non-homogeneous Poisson-process model for software reliability. IEEE Trans. Reliabl. 42, 604–612.
13. Jelinski, Z., Moranda, P.B., 1972. Software reliability research. In: Freiberger, W. (Ed.), Statistical Computer Performance Evaluation. Academic Press, New York, pp. 465–484.
14. Kaufman, L.M., Dugan, J.B., Johnson, B.W., 1998. Using statistics of the extremes for software reliability analysis of safety critical systems. In: Proc. Ninth Int. Symp. Software Reliab. Eng., pp. 355–363.
15. Kaufman, L.M., Dugan, J.B., Johnson, B.W., 1999. Using statistics of the extremes for software reliability analysis. IEEE Trans. Reliabl. R-48, 292–299.
16. Knafl, G., Morgan, J., 1996. Solving ML equations for 2-parameter Poisson-process model for ungrouped software failure data. IEEE Trans. Reliabl. 45, 42–53.
17. Langberg, N., Singpurwalla, N.D., 1985. Unification of some software reliability models. SIAM J. Sci. Comput. 6 (3), 781–790.
18. Lyu, M.R. (Ed.), 1996. Handbook of Software Reliability Engineering. McGraw-Hill, New York.
19. Miller, D.R., 1976. Order statistics poisson processes and repairable systems. J. Appl. Probab. 13, 519–529.
20. Miller, D.R., 1986. Exponential order statistic models of software reliability growth. IEEE Trans. Software Eng. SE-12, 12–24.
21. Musa, J.D., 1999. Software Reliability Engineering. McGraw-Hill, New York.
22. Musa, J.D., Iannino, A., Okumoto, K., 1987. Software Reliability, Measurement, Prediction, Application. McGraw-Hill, New York.
23. Ohba, M., 1984. Software reliability analysis.IBM. J. Res. Develop. 28, 428–443.
24. Ohba, M., 1984. Inflection S-shaped software reliability growth model. In: Osaki, S.