LOGISTIC REGRESSION APPROACH TO SOFTWARE RELIABILITY ENGINEERING WITH FAILURE PREDICTION

K.Venkata Subba Reddy [1] and Dr.B.Raveendra Babu[2]

[1]Assistant Professor, Department of Computer Science and Engineering
Muffakham Jah College of Engineering and Technology, Hyderabad, India.
kvsreddy2012@gmail.com
[2]Director, Vignana Jyothi Institute of Management, Hyderabad, India.
rbhogapathi@yahoo.com

ABSTRACT

In this paper using the main feature of our proposed Model in its inflection point, we propose a software reliability growth model, which relatively early in the testing and debugging phase, provides accurate parameters estimation, gives a very good failure behavior prediction and enable software developers to predict when to conclude testing, release the software and avoid over testing in order to cut the cost during the development and the maintenance of the software. Two real world experimental data previously analyzed have been used to compare our proposed Early Estimation Logistic Model effectiveness with several pre-existing models.

KEYWORDS

Imperfect Debugging, Logistic Regression, Software Reliability

1. INTRODUCTION

The increasing dependency on computers along with an increasing need for safety, efficiency and cost reduction, causes not only for a higher demand for software and make software testing a crucial phase of the development of any software to produce highly reliable packages. It is clear that air traffic control software will demand a very accurate failure behaviour prediction, and then software designed for a fun game. A crucial decision any software developer has to make, in software testing and debugging phase, is the following: given all the restrictions, can we stop the testing and debugging phase and release a software package for use, or should we continue testing?

Since the occurrence of software failures are random, if the testing and debugging team hits a long random run where no software mistake is found; they are likely to stop testing and falsely conclude that the software is error free. This leads to a premature end of the testing and debugging phase that affect the reliability of the software to be released for marketing. Satoh and Yamada's software reliability growth models do not yield accurate reliability estimates in spite of a small amount of failure data causing their procedure and justification do not lead to a good estimation at the early stage of the testing and debugging phase. In this paper, we propose a more realistic model that is more effective and eliminates many of the difficulties that the S-Y models require. Our proposed software reliability growth model is simple to implement and does not assume any prior distribution. Relatively early in the testing and debugging phase, it provides accurate parameters estimation, gives a very good failure behaviour prediction and it is possible

DOI : 10.5121/ijsea.2013.4105

for software developers to predict when to conclude testing, release the software and avoid over testing in order to cut the cost during the development and the maintenance of the software.

2. STATISTICAL ABREVIATIONS AND NOTATIONS

(i) . LR-EE-Model: our Logistic Regression Early Estimation Model
(ii). S-Y Models: Satoh and Yamada's Software Reliability Growth Models

3. PRELIMINARIES

Here, we shall give a brief description of Satoh and Yamada statistical models.

3.1 Satoh and Yamada's models and conclusions

Satoh and Yamada reported that, their two software reliability growth models yield accurate parameter estimates even with a small amount of data, [1]. Their proposed models give the same parameters estimates. According to Satoh and Yamada, although the conventional model uses a discrete equation as a regression model, the model itself is a continuous time model, thus, it includes errors generated by discretization, however, their models do not have this problem because they are themselves discrete models and that they can analyze the software reliability without a continuous time model.

The Satoh and Yamada discrete models were reported to have three advantages over the conventional model [1].

(1). The parameter estimation in the discrete logistic curve models reproduced the values of the parameters very accurately, even with small data that do not include the inflection point. When the exact solution is used as the input data, the conventional model provides inaccurate parameter estimates with data that do not include the inflection point.

(2). The discrete models are independent of time scale. The same parameters estimates are obtained no matter what value of the time scale they choose. When the conventional model is used we have to choose the time scale carefully, because the time scale needs to be used in the regression equation. As a result, the estimates depend on the choice of the time scale.

(3). The discrete logistic curve models enable them to accurately estimate the parameters in the early testing phase with real data.

The parameter estimates of the conventional model vary with the number of data points. The discrete models provide stable values of parameter estimates for various number of data points. This characteristic is very important for software reliability growth models.

3.2 Remarks

We believe that Satoh and Yamada's conclusion results from not simulating the data properly; they use the exact solution is used as the input data not adding any noise. As a result, the problem is purely deterministic. Since there are three parameters k, m, and α to estimate in

$L(t) = \frac{k}{1 + m\exp(-\alpha t)}$, if no noise is added to an exact solution when preparing the data, as a Saturated model one only needs as many data points as there are parameters to estimate the parameters. While Satoh and Yamada's approach perfectly reproduces the values of the parameters, when the data set satisfy an exact solution with no noise, in early stage of the

debugging and testing phase (three data points), their parameter estimation with too small data points does not give accurate parameter estimates on prepared data with noise and on actual data.

4. DEVELOPMENT OF THE PROPOSED MODEL : EARLY ESTIMATION

The crucial decision any software developer have to make, in software testing and debugging phase is whether to stop, conclude the testing and the debugging phase of a software or continue testing and debugging it. Having to take into consideration factors like reliability of the software being developed and the cost of its development process, it is of gold for the software developers to have a stopping rule, at the early stage of the testing and debugging phase, associated with a high confidence level that testing and debugging phase must be concluded at some time t¤, to reduce cost of over testing the software or that testing must proceed a bit longer to avoid a premature ending of the debugging process to produce more reliable or highly reliable software.

The objective of this section is to develop a software reliability growth model that

(i). Does not assume any prior distribution

(ii). is free of any major convergence assumptions

(iii). is simple

(iv). fits very well the cumulative number of software faults found and corrected

(v). Provides accurate parameter estimates relatively early in the testing phase once the available data points during testing and debugging phase include a bit more then the inflection point.

(vi). Our proposed model provides stable values of parameter estimates once the available data includes a bit more then the inflection point; this characteristic is very important for software reliability growth models.

(vii). Provides an additional decision-making rule to the software developers respective to when to conclude the testing phase, manage their resources, and schedule the release time of the software package to be marketed for use.

(viii). Not subject to errors generated by discretization.

(ix).our LR-EE-Model, by providing a stopping and release rule, saves time and cuts cost in preventing over-testing the software to be marketed, and produces a more reliable software in preventing an early release of the software to be market, specially when the testing and debugging team hits a long run where not software fault is found.

4.1 Model Description

The proposed model is dependent on an accurate estimated number of software faults in the software $\hat{k}$, before complete modelling of the cumulative failure behaviour is obtained. In this chapter, we propose a new way of estimating the parameter k, as an increasing number of available failure data are recorded, while the testing and debugging phase proceeds. Our new estimate of the parameter k is based on an accurate method to locating the inflection point of a gradually increasing amount of actual failure data points. Once the inflection point is located, a

prediction of the total number of software faults $\hat{k}$ is obtained, then estimation of the remaining parameters m and n is obtained.

5. PARAMETERS ESTIMATION

Clearly, if k the initial number of software faults in the software prior the debugging phase or the total number of software faults found and corrected in debugging phase during an infinitely long exposure time interval, when a Cumulative logistic curve model is fitted, at the inflection point k/2 software faults have been found and corrected. In this section, we propose a method for locating the inflection point; then an estimate of k is found via estimating k/2.

Let $T_1 < T_2 < T_3 <\ldots\ldots < T_n$ be the first n successive random failure times, as software ages, with their corresponding cumulative number of software faults found and corrected:

$L(T_1) = 1, L(T_2) = 2, L(T_3) = 3 , ..., L(T_n) = n.$

We run sequential linear regression on two kind of successive overlapping chains of size N ordered pairs (T_i, L (T_i) = i) that we will refer to as Chain1 or Chain2.

5.1. Description of overlapping chain 1

Our chain *I* consists of overlapping chains of size $N \geq 2$ constructed as follows: Our first run consist of finding the equation of the regression on the first *N* ordered pairs (T_i , $L(T_i) = i$), $i = 1,\ldots\ldots,N$ and recording its slope, then for any other successive k^{th} run $k \geq 2$, we run a linear regression line on (T_i , $L(T_i) = i$), where $i = (k_{i1})(N_{i1})$; i + 2, i + 3, ……, i + N.

5.2. Description of overlapping chain 2

Our chain II consists of overlapping chains of size N≥4 and N is even, constructed as follows:
Our k^{th} run consists of finding the equation of successive regressions lines on the ordered pairs (T_i, $L(T_i) = i$), where $i = (k\text{-}1)(N/2) + 1$; $i + 1, i + 2,\ldots\ldots, i + (N_{i1})$.

5.3. Locating the inflection point

For both successive overlapping chains of size N ordered pairs (T_i, $L(T_i) = i$), the slopes of the successive regression lines are recorded and their patten is observed. To locate the inflection point, we look for the chain on which the successive slope behaviour is maximal then consistently decreasing. Once the chain including the inflection point is identified, averaging is used to estimate *k*.

(1) Select using the maximum slope criteria an i^{th} chain
(2) In the selected i^{th} chain, two times the actual cumulative faults is a candidate for $\hat{k}$.
(3) Test each candidate for $\hat{k}$ in the selected i^{th} chain, using the minimum mean square error criteria:

Let N, M, and three candidates from the selected i^{th} chain such the N < M < L If MSEN >MSEM, discard candidate N and proceed with comparing the mean square errors MSEM and MSEL.

If *MSEN < MSEN*, select candidate *N* as the estimate of the parameter *k*.After estimating *k*, we are ready to set up our Logistic Like Software Reliability Growth Model.

Having an estimate of *k*, we name it $\hat{k} = i_{early}$, we set up our best fit logistic like regression model in the minimal mean square error sense.
Proposing

$$\overline{P(t)} = \frac{e^{\hat{\beta}_{0\,i_{early}}} + e^{\hat{\beta}_{1\,i_{early}}\,t}}{1 + e^{\hat{\beta}_{0\,i_{early}} + \hat{\beta}_{1\,i_{early}}\,t}}$$

to estimate the true cumulative percentage number of software failures found and corrected during debugging and testing phase up to testing and debugging time *t*, we obtain estimates of *m* and α needed in Equation (1) as follows:

By identification method we obtain

$$\frac{1}{1+me^{-\alpha t}} \equiv \frac{e^{\hat{\beta}_{0i_{early}}} + e^{\hat{\beta}_{i_{early}}t}}{1+e^{\hat{\beta}_{0i_{early}} + \hat{\beta}_{i_{early}}t}} = \frac{1}{1+\frac{1}{e^{\hat{\beta}_{0i_{early}}}\,\hat{\beta}_{1i_{early}}t}} = \frac{1}{1+e^{-\hat{\beta}_{0i_{early}} - \hat{\beta}_{1i_{early}}t}} \qquad \textbf{(1)}$$

Now comparing the first and the last term of Equation 1, we obtain the following estimates of *m* and α:

$$\hat{m} = e^{-\hat{\beta}_{0i_{early}}}$$

$$\hat{\alpha} = \hat{\beta}_{1_{i_{early}}}$$

Where $\hat{\beta}0i_{early}$ and $\hat{\beta}1i_{early}$ are the Maximum Likelihood Estimators of the parameters $\hat{\beta}0$ and $\hat{\beta}1$, using $\hat{k} = i_{early}$ and i_{early} the positive integer from *N, M, L* at which slope is maximal and mean square error is minimal. Thus, an estimate of *P*(*t*) can be obtained using

$$\overline{P(t)} = \frac{1}{1+me^{-\hat{\alpha}t}} = \frac{1}{1+e^{-\hat{\beta}_{i\,iearly}}e^{-\hat{\beta}_{1\,iearly}t}}$$

Finally the cumulative failure behaviour of the proposed model for given software is given by

$$\overline{L(t)} = \frac{\hat{k}}{1+\hat{m}e^{-\hat{\alpha}t}} = \frac{i_{early}}{1+e^{-\hat{\beta}_{0i_{early}}}e^{-\hat{\beta}_{1i_{early}}t}}$$

It will be shown that our proposed estimate of *L*(*t*) gives good results in comparison to the other models that are commonly used.

6. COMPARISIONS OF MODELS : NUMERICAL APPROACH TO SOFTWARE FAILURE DATA

In this section we shall illustrate our approach and compare the performance of our Early Estimation Logistic Model, using overlapping chains, and using the mean square error, with several frequently and pre-existing models on an actual software failure data already analyzed in the PL/I application program test data Table 1, reported and studied by Ohba [10] in 1984.

It will be shown that our proposed model yields accurate parameters estimates early into the testing and debugging phase, gives a very good modelling of the S-shaped cumulative number of software faults found and corrected curves during testing and debugging phase giving the software developers an early protocol as far as when to stop the testing and debugging phase and release the software for use, cutting cost on maintenance, and due to over testing.

6.1. Our Early Estimation on the PL/I Software Failure Data

From Tables 1, Table 2, Table 3, and Table 4, respectively we estimated a cumulative number of software fault of 350. Thus, our estimate $\hat{k}$ = 350. Table 5, provides a comparative summary of our estimate $\hat{k}$ along with several other estimates from some pre-existing models and the mean square error for each model. Table5 also shows that by predicting 350 faults associated with an overall mean square error of 92*:*31760154 our proposed model, LR-Model, without any major assumptions, fitting well trough the K-S goodness of-fit, demonstrates better performance than most of the pre-existing models when data points a bit after the inflection points are available.

From Tables 1, Table 2, Table 3, and Table 4, respectively we estimated a cumulative number of software fault of 350. Thus, our estimate $\hat{k}$ = 350.Table 5, provides a comparative summary of our estimate $\hat{k}$ along with several other estimates from some pre-existing models and the mean square error for each model. Table5 also shows that by predicting 350 faults associated with an overall mean square error of 92*:*31760154 our proposed model, LR-Model, without any major assumptions, fitting well trough the K-S goodness of-fit, demonstrates better performance than most of the pre-existing models when data points a bit after the inflection points are available.

6.2. Creating Overlapping Chains: PL/I Database Application Software

Table 1: Overlapping Chain 1 of Size N = 2 -PL/I Database Application Software

RUNS	$[T_i,T_{(i+1)}][N_i,N_{(i+1)}]$	Slope of RL	r
RUN1	[1,3];[15,66]	25.5	1
RUN2	[2,4];[44,103]	29.5	1
RUN3	[3,5];[66,105]	19.5	1
RUN4	[4,6];[103,110]	3.5	1
RUN5	[5,7];[105,146]	20.5	1

RUN6	[6,8];[110,175]	32.5	1
RUN7	[7,9];[146,179]	16.5	1

Note that only the data points up to the 9^{th} week are use to predict the total number of faults in the software.

Table 2: Overlapping Chain 1 of Size N = 2 -PL/I Database Application Software

N	MSE	Pair number
N=88	6.61E-09	2
N=132	15.30759212	3
N=206	16.17930627	4
N=220	179.7683445	5
N=292	179.2839406	6
N=350	**165.1157652**	**7**
N=351	165.259913	8
N=358	183.1532424	9
N=412	189.5550172	10
N=466	189.8825614	11

Table 3: Overlapping Chain -I of Size N = 3-PL/I Database Application Software

RUNS	$[T_i,T_{(i+1)}][N_i,N_{(i+1)}]$	Slope of RL	r
RUN1	[1,2,3];[15,44,66]	25.5	r:=0.9968748929
RUN2	[2,4,5];[44,103,105]	21.64285714	r:=0.9539628928
RUN3	[4,6,7];[103,110,146]	12.78571429	r:=0.8464894644
RUN4	[6,8,9];[110,175,179]	24.35714286	r:=0.9605519477
RUN5	[8,10,11];[175,206,233]	18.78571429	r:=0.9887218045

Note that only the data points up to the 11^{th} week are use to predict the total number of faults in the software.

Table 4: Overlapping Chain-I of Size *N* = 5-PL/I Database Application Software

RUNS	$[T_i,T_{(i+1)}][N_i,N_{(i+1)}]$	Slope of RL	r
RUN1	[1,2,3,4,5];[15,44,66,103,105]	23.9	r:=0.9778998350
RUN2	[4,6,7,8,9];[103,110,146,175,179]	17.33783784	r:=0.9416620919
RUN3	[8,10,11,12,13];[175,206,233,255,276]	20.60810811	r:=0.9952196625

Note that only the data points up to the 13^{th} week are use to predict the total number of faults in the software.

Table 5: Summary of models estimations: PL/I Database Application Software

Models	**a or k**	**MSE**	**AE**
Our early estimation model	350	92.31760154	2.23
Our LR-Model	348	91.92380892	2.79
HLM Model Group A, with Logistic function	394.076	118.29	10.06
HLM Model Group A, with Weibull function	565.35	122.09	57.91
HLM Model Group A, with Rayleigh function	459.08	268.42	28.23
HLM Model Group A, with Exponential	828.252	140.66	131.35
HLM Model Group B with Logistic function	337.41	163.095	5.75
HLM Model Group B, with Weibull function	345.686	91.0226	3.43
HLM Model Group B, with Rayleigh function	371.438	158.918	3.75
HLM Model Group B, with Exponential	352.521	83.998	1.53
HLM Model Group C, with Logistic function	430.662	103.03	20.11
HLM Model Group C, with Weibull function	385.39	87.5831	7.65
HLM Model Group C, with Rayleigh function	379.947	406.71	6.13
HLM Model Group C, with Exponential	385.179	83.3452	7.69
HLM Model Group D, with Logistic function	582.538	96.9321	62.72
HLM Model Group D, with Weibull function	958.718	124.399	167.79

HLM Model Group D, with Rayleigh function	702.693	247.84	96.09
HLM Model Group D, with Exponential	1225.66	169.72	242.36
G-O Model	562.8	157.75	56.98
Inflection S-Shaped Model	389.1	133.53	8.69
Delayed S-Shaped Model	374.05	168.67	4.48
Exponential Model	455.371	206.93	27.09
HGDM	387.71	138.12	8.3
Logarithmic Poisson Model	NA	171.23	

7. CONCLUSIONS

Modelling accurately the cumulative number of software faults *k* in a software package is crucial to software developers. In the present study we have introduced a logistic software reliability growth model for the purpose of early estimation of the cumulative number of faults in given software. The analytical form of this new model is given by

$$\overline{L(t)} = \frac{\hat{k}}{1+\hat{m}e^{-\hat{\alpha}t}} = \frac{i_{early}}{1+e^{-\hat{\beta}_{0i_{early}}}e^{-\hat{\beta}_{1i_{early}}t}} \quad \text{With} \quad \hat{m} = e^{-\hat{\beta}_{0i_{early}}}$$

$$\hat{\alpha} = \hat{\beta}_{1_{i_{early}}}$$

Where $\hat{\beta}0i_{early}$ and $\hat{\beta}1i_{early}$ are the Maximum Likelihood Estimators of the parameters $\beta 0$ and β1, using $\hat{k} = i_{early}$ and i_{early} the positive integer candidate at which slope is maximum and at which $\overline{MSE}$ is minimal as the number of pairs increases. This model not only gives very good predictions of the cumulative number of software faults, but it is easy to apply and it is free of any major assumptions. The new model was compared with the following commonly used models in the subject areas. (i) Exponential Model, (ii) G-O Model, (iii) Delayed S-Shaped Model, (iv) S-Y Models,(v) C-Model,(vi)Hyper geometric Model, (vii) Huang, Kuo, Chen, and Lyu's Models [5][6][7][9].

For a first application, we used one set of actual data, namely, the PL/I Database Application test data by Ohba, [10].Later we will apply our early estimation to the remaining sets analysed earlier. (i) The pattern of discovery of errors test Data by Tohma, [11]. (ii) The F 11-D program test data by Forman and Singpurwalla, [13] (iii) The pattern of discovery of errors STS2, STS3, STS4 by Misra [15] (iv) Musa's System T1 [17].(v) The On-line data entry software package test Data by Ohba,[10].(vi) A field report test Data by Tohma, [11][16][18]. The mean square error criteria were used to compare the results of our proposed model with other models stated above. The results presented in tables form support the fact that the new model is more effective in estimating the number of faults found during the testing and debugging phase of given software.

.

REFERENCES

[1] D. Satoh and S.Yamada, "Parameter Estimation of Discrete Logistic Curves Models for Sofware Reliability Assessement," Japan J. Indust. Appl. Math., 19(2002); 39 ¡ 53

[2] F. Morishita, "The fitting of the logistic equation to the rate of increase of population density," Res. Popul. Ecol. V II (1965); 52 ¡ 55

[3] "Computer science and statistics: proceedings of the Sixteenth Symposium on the Interface," Atlanta, 1984.

[4] D. W. Hosmer and S. Lemeshow, "Applied logistic regression," John Wiley and Sons, NY,1989.

[5] C. Y., J. H. Lo, and S.Y. Kuo, "Pragmatic study of Parametric Decomposition Models for Estimating Software Reliability Growth," Proc. 9th Int'l. Symp. Software Reliability Engineering (ISSRE'98), 1998, pp. 111 ¡ 123.

[6] C. Y. Huang, S. Y. Kuo, and I. Y. Chen, "Analysis of a Software Reliability Growth Model with Logistic Testing-Effort Function," Proc. 8th Int'l. Symp. Software Reliability Engineering (ISSRE'97), 1997, pp. 378 ¡ 388.

[7] C. Y. Huang, S. Y. Kuo, and M. R. Lyu, "Effort-Index-Based Software Reliability Growth Models and Performance Assessment," Proc. 24th Ann. Int'l. Computer Software and Applications Conf. (COMPSAC 2000), 2000.

[8] R. H. Hou, S. Y. Kuo, and Y. P. Chang, "Applying Various Learning Curves to Hyper-Geometric Distribution Software Reliability Growth Model," Proc. 5th Int'l. Symp. SoftwareReliability Engineering, 1994, pp. 7 ¡ 16. 107

[9] S. Y. Kuo, C. Y. Huang, and M. R. Lyu, "Framework for Modeling Software Reliability, Using Various Testing-Efforts and Fault-Detection Rates," IEEE TRANSACTIONS ON RELIABILITY, VOL. 50, NO. 3, SPTEMBER 2001.

[10] M. Ohba, "Software reliability analysis models," IBM J. Res. and Development, vol. 28, no. 4, pp. 428 ¡ 443, Jul. 1984.

[11] Y. Tohma , R. Jacoby, Y. Murata, and M. Yamamoto, "Hyper-Geometric Distribution Model to Estimate the Number of Residual Software Faults," Proc. COMPsac-89, Orlando, pp. 610¡617, September 1989.

[12] Y. Tohma, H. Yamamoto and R. Jacoby, "Parameters Estimation of the Hyper-Geometric Distribution for Real/Test Data," Proc. of the 18th International Symposium on Fault Tolerant Computing, pp. 148 ¡ 153, 1988, Japan.

[13] E. H. Forman and N. D. Singpurwalla, "An Empirical Stopping Rule for Debugging and Testing Computer Software," J. American Statistical Association, Vol. 72, December 1977, pp. 750 ¡ 757.

[14] C. J. Dale and L. N. Harris: "Approach to Software Reliability Prediction," Proc. Ann. Reliability and Maintainability Symposium, pp. 167 ¡ 175, 1982.

[15] P. N. Misra: "Software reliability analysis," IBM Systems Journal, Vol 22, NO 3, 1983.

[16] Y. Tohma, H. Yamano, M. Ohba, and R. Jacoby, "The Estimation of Parameters of the Hypergeometric Distribution and Its Application to the Software Reliability Growth Model," IEEE, 1991.

[17] T. Doli, N. Wakana, S. Osaki, and K. S. Trivedi, "Analysis of Hypergeometric Distribution,Software Reliability Model," IEEE, 2001.

[18] Y. Tohma, K. Tokunaga, S. Nagase, and Y. Murata, "Structural Approach to the Estimation of the Number of Residual Software Faults Based on the Hyper-Geometric Distribution," IEEE, 1989.

Short Biography

[1]***K.Venkata Subba Reddy*** obtained his B.Tech in Information Technology from University of Madras in 2002 and received the M.Tech in Software Engineering from Bharath University, Chennai in 2005, He is pursuing Ph.D., in Computer Science and Engineering, under the guidance of Dr.B.Raveendra Babu, at Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. He is currently working as Assistant Professor in Computer Science & Engineering Department at Muffakham Jah College of Engineering & Technology, Banjarahills, Hyderabad. He has 9 years of experience in teaching. His research interests include Software Engineering, Software Reliability Engineering and Cloud computing. He is a life member of ISTE and a member of CSI.

[2]***Dr B. Raveendra Babu*** obtained his Masters in Computer Science and Engineering from Anna University, Chennai. He received his Ph.D. at S.V University, Tirupati. He is currently leading a Team as Director Vignana Jyothi Institute of Management, Hyderabad. He has 26 years of teaching experience. He has more than 25 international & national publications to his credit. His research areas of interest include Software Engineering, Image Processing, and Information Security.