Application of Goel-Okumoto Model in Software Reliability Measurement

Pankaj Nagar
Department of Statistics, University of Rajasthan, Rajasthan, India
pnagar121@gmail.com

Blessy Thankachan
Department of Statistics, University of Rajasthan, Rajasthan, India
blessy218@gmail.com

ABSTRACT
The estimation of remaining errors in the software is the deciding factor for the release of the software or the amount of more testing which is required. Software growth reliability models are used for the correct estimation of the remaining errors. In this paper the Goel-Okumoto Model has been selected and its various parameters are discussed with a case study. A criterion has also been evaluated for the estimation of reliability of any software.

Keywords-
Calendar time, Residual Errors, Reliability Factor, Roundness Factor

1. INTRODUCTION
The probability that software will work and produce desirable outputs for a specified time under a certain environment is called the reliability of the software. Numerous methods have been designed which can help in improving the reliability of the software before it is shipped to the user. To make reliable software intensive and careful planning of testing phase and accurate decision-making is required. This careful planning and decision-making requires the use of software reliability analysis model or software reliability growth model. Software Reliability Growth Models usually have the form of random process that describes the behavior of failures with respect to time. It specifies the general form of the dependence of the failure process on the principle factors that affect it: fault introduction, fault removal, and the operational environment i.e Software reliability modeling is done to estimate the form of the failure rate function by statistically estimating the parameters associated with a selected mathematical model. At any particular time it is possible to observe a history of the failure rate (failures per unit time) of software. Fault identification and removal generally force the failure rate of a software system to decrease with time. The purposes of modeling are:

- To estimate the remaining time required to achieve a specified objective.
- To estimate the expected reliability of the software when the product is released.

Measurement of software reliability comprises of the determination of software reliability or its alternative quantities from defect data. Software reliability growth models have many underlying assumptions that are often violated in practice, but empirical evidence has shown that many are quite robust despite these assumption violations [1]. Because of assumption violations, it is often difficult to know which models to apply in practice. The model presented here is the basic execution time model or the Goel-Okumoto Model [2].

2. GOEL-OKUMOTO SOTWARE RELIABILITY GROWTH MODEL
The primary objective of a software reliability model is to forecast failure behavior of the software that will be experienced when the software is operational. This expected behavior changes rapidly and it can be tracked during the period in which the program is tested.

A. Basic Assumptions of Goel/Okumoto Model

- The execution times between the failures are exponentially distributed.
- The cumulative number of failures follows a Non Homogeneous Poisson process (NHPP) by its expected value function μ(t).
- For a period over which the software is observed the quantities of the resources that are available are constant.
- The number of faults detected in each of the respective intervals is independent of each other.
- The mean value function is such that the expected number of error occurrences for any time t to t+Δt is proportional to the expected number of undetected errors at time t. It is also assumed to be bounded, non-decreasing function of time with $\lim_{t\to\infty} \mu(t) = N < \infty$
- Fault causing failure is corrected immediately; otherwise reoccurrence of that failure is not counted (repair is immediate and perfect).

B. Goel-Okumoto Baisc Model

$\mu(t) = E_E (1-e^{-bt})$, where $E_E \geq 0, b>0$ (1)

$\mu(t)$ = Predicted number of defects at time t

E_E = Expected total number of defects in the code in infinite time (it is usually finite)

b = Roundness factor/shape factor = the rate at which the failure rate decreases.

t = Calendar time/ execution time/ number of test runs

Because it is a non-linear equation, the solution found may be local optimum rather global optimum. Therefore it is beneficial to define parameter values that are close to the final values [3]. The parameter values which are selected should provide a reasonable match to the existing data. But it is worthy only if the estimation is done already, or the analysis is done after the software is released to the user. If the result is obtained using the previous month's data, those parameter values are a good starting point to estimate the value of expected total no. of defects and the roundness factor.

C. Components of the Model

1) Expected no. of Defects: In this model $\mu(t)= E_E F(t)$. Here F(t) is a cumulative distribution function. F(0)=0, i.e number of defects are 0 before the test starts, and $F(\infty)=1$, therefore $\mu(\infty)=E_E$ and E_E is the total number of bugs detected after an infinite number if testing is done.

This model attempt to statistically correlate defect detection data with other known functions like exponential functions. The model have a parameter that relates to the total number of bugs contained in the entire cod. Residual Errors [3] can be found out if the entire no. of bugs is detected and calculated as follows:

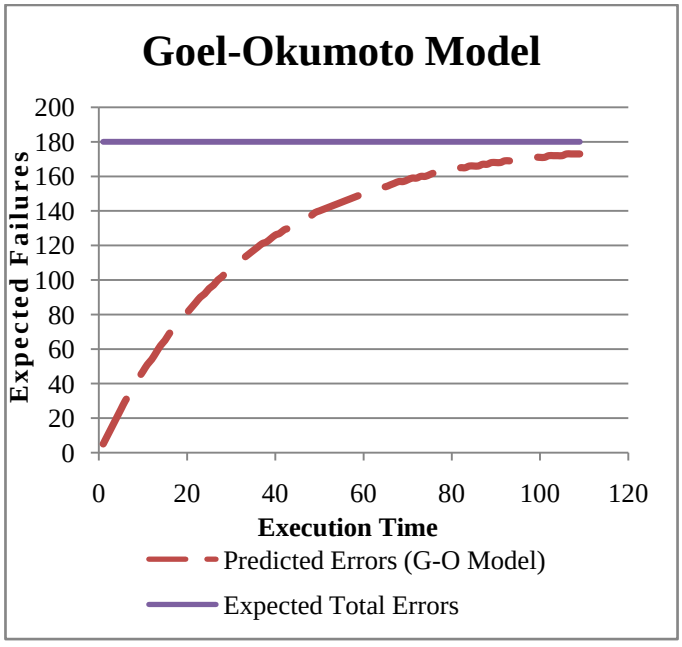
Residual Errors = Total number of errors in the code - errors discovered and rectified.

2) Roundness Factor: The roundness factor for a perfect circle has the value '1' and for shapes with increasing irregularity, the value tends to '0'. Other shape factors are sensitive especially for the presence of concave irregularities, whereas factors like the roundness factor can have the same value for shapes with many small concave irregularities and for elongated shapes without concave irregularities [4].

3) Test Time Data: For any software reliability growth model, the appropriate measure of time must relate to the testing effort. There are three possible methods for measuring test time:

- Calendar time
- Number of tests run
- Execution (CPU) time.

Plot of Expected Failure in Goel-Okumoto is shown:



D. Estimation of Model Parameters

In the case of the above model, two parameters must be estimated: total expected failures for infinite time (E_E) and the rate of reduction in the failure rate or the roundness factor (b).

The parameters can be detected during two phases:

- During the testing phase or before the software is shipped to the client. Statistical inference methods like Maximum Likelihood, Classical Least Square, and Alternative Least Square can be used to estimate the parameters in terms of calendar time.
- If the predictions are done after the software is shipped to the client, then it is done through software characteristics like size and complexity of the software or the failure data. Once the failure data is available in terms of execution time, these parameters may be estimated, using any statistical inference method [5]. The accuracy of the parameters generally increases with the size of the sample of failures.

3. ESTIMATION OF SOFTWARE RELIABILITY

The Measurement of software reliability involves estimation of software reliability or the alternate quantities of software reliability from failure data. Typically, software reliability prediction takes into account factors such as the size and complexity of a program [5]. Reliability of the software or MTTF increases as a function of execution time. Reliability can be measured on the basis of error detection rate per unit time.

These predictions can be further used to analyze the reliability of the software.

$E_R = E_E - \mu(t_{last})$, where, (2)

E_R = Residual Errors

E_E = Total Expected Errors

$\mu(t_{last})$ = Predicted Errors at the end of quality assurance period

(Goel-Okumoto Model)

Reliability Factor (RF) = $1-(E_R / E_E)$ (3)

Reliability Factor (RF) is the measure for software reliability. Its value varies between 0 and 1. If RF=1, then the software under consideration is perfect, however, if RF=0, then the software is highly vulnerable. When RF approaches close to 1 then the software can be considered as reliable.

4. CASE STUDY

In this section, we show a demonstration of the tool using a data set. It is assumed that the software is in active use for not less than 10 years. This whole period is considered as the life of the software during which defects are being discovered and rectified continuously. As the defects are being discovered and removed the number of residual errors (E_R) remaining in the code goes on decreasing and thus increasing the reliability factor (RF). The data set, reproduced in Table 1, consists of 10 observations corresponding to times between testing. The total expected defects (E_E) in the code are 100. The roundness factor or the defect reduction rate is considered to be between 0.03 and 0.05 (based on empirical studies of several softwares). The value of the roundness factor b depends upon the type of software and the environment in which it is being used.

In the initial year the number of residual errors left in the code is high but during the lifetime of the software the residual errors keep on decreasing, which signifies that the errors lefts in the code approaches 0. Thus the reliability factor (RF) also approaches 1,

signifying that the reliability of the software under consideration tends to improve as the software is being used continuously for considerable amount of time.

E_E	T (In Months)	B	$\mu(t_{last})$	E_R	RF
100	12	0.05	45.11542	54.88458	0.451154
100	24	0.05	69.87683	30.12317	0.698768
100	36	0.05	83.46703	16.53297	0.83467
100	48	0.05	90.92595	9.074053	0.909259
100	60	0.05	95.01974	4.980256	0.950197
100	72	0.05	97.26661	2.733392	0.972666
100	84	0.05	98.49979	1.500211	0.984998
100	96	0.05	99.17662	0.823384	0.991766
100	108	0.05	99.54809	0.451911	0.995481
100	120	0.05	99.75197	0.248029	0.99752

Table 1- Estimation of RF when b=0.05

E_E	t(Months)	b	$\mu(t_{last})$	E_R	RF
100	12	0.04	38.11858	61.88142	0.381186
100	24	0.04	61.7069	38.2931	0.617069
100	36	0.04	76.30369	23.69631	0.763037
100	48	0.04	85.33638	14.66362	0.853364
100	60	0.04	90.92595	9.074053	0.909259
100	72	0.04	94.38485	5.615153	0.943848
100	84	0.04	96.52526	3.474736	0.965253
100	96	0.04	97.84978	2.150216	0.978498
100	108	0.04	98.66942	1.330584	0.986694
100	120	0.04	99.17662	0.823384	0.991766

Table 2- Estimation of RF when b=0.04

E_E	t (Months)	b	$\mu(t_{last})$	E_R	RF
100	12	0.03	30.22976	69.77024	0.302298
100	24	0.03	51.32114	48.67886	0.513211
100	36	0.03	66.03664	33.96336	0.660366
100	48	0.03	76.30369	23.69631	0.763037
100	60	0.03	83.46703	16.53297	0.83467
100	72	0.03	88.4649	11.5351	0.884649
100	84	0.03	91.95194	8.048063	0.919519
100	96	0.03	94.38485	5.615153	0.943848
100	108	0.03	96.08229	3.917705	0.960823
100	120	0.03	97.26661	2.733392	0.972666

Table 3- Estimation of RF when b=0.03

Thus it can be understood that the value of RF is the deciding factor for the estimation of reliability of software. Typical values of RF for reliable softwares should be:

S. No.	Type of S/W	Expected value of RF
1.	Platform Software [6]	$0.95 \leq RF \leq 1.0$
2.	Application Software [6]	$0.90 \leq RF \leq 0.95$
3.	User Written Software [6]	$0.80 \leq RF \leq 0.90$

Table 4 – Expected RF values for different types of softwares

4. CONCLUSION

Software reliability growth models can provide a good prediction of the number of failures at time T and thus the number of remaining failures (residual errors) can be found out. It is seen that the basic execution model is generally superior in capability and applicability to the other published models [5]. Wood's empirical study [7] has shown that predictions from simple models of cumulative defects based on execution time correlate well with field data [1]. In our study, predictions from the Goel-Okumoto model based on calendar time correlate well with data from our environment. The model provides the reasonable estimation of predicted errors. Reliability factor (RF) can be considered a good criterion for the analysis of the reliability. The correct estimation of RF value however depends upon the correct estimation of Total Expected Errors (EE) and the Roundness Factor (b). Roundness factor depends upon the rate of decrease of errors or the rate in which the errors are being detected and recovered. When the software is high risk software the testing efforts would be too high thus leading to fast retrieval of errors, and thus the rate of detection of errors would be high and the value of b would approach 1, and reliable software can be produced in less time and vice versa.

REFERENCES

[1] C Stringfellow, A Amschler Andrews “An empirical method of selecting software reliability growth models”, Empirical Software Engineering, 7, 319–343, 2002.2 Kluwer Academic Publishers. Manufactured in The Netherlands.

[2] J.D.Musa, K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement", Proc. 7th International Conference o_nn Software Engineering, Orlando, Florida, March 26-29, 1984, pp. 230-238.

[3] Alan Wood “Software Reliability Growth Models”, Technical Report, Part Number 130056, September 1996

[4] Reinhold Nafe 1 , Wolfgang Schlote, “Methods for Shape Analysis of two-dimensional closed Contours - A biologically important, but widely neglected Field in Histopathology” Electronic Journal of Pathology and Histology Volume 8.2; June 2002

[5] John D Musa, Kazuhira Okumoto “Application of basic and logarithmic poisson execution time models in software reliability measurement”, Proceeding Software Reliability Modelling and Identification, Springer-Verlag London, UK ©1988, ISBN:3-540-50695-0

[6] http://en.wikipedia.org/wiki/Software A. Wood, “Predicting Software Reliability," IEEE Computer, vol. 29, no. 11, (November 1996), pp. 69 78.