Volume 4, Issue 4, April 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Study of Various Reliability Growth Models

Sonia Deswal[1]
[1]*Department of Computer Science*
PDM College of Engineering, MD University
Bahadurgarh, HR, India

Renu Dalal[2]
[2]*Department of Computer Science*
AIACT&R, IP University
Geeta Colony, Delhi-110031, India

Abstract -In literature, we have various software reliability growth models (SRGM) which have been developed to facilitate the developers in monitoring the reliability of the software during the software development. Software reliability models can be used to predict the behaviour of software systems. SRGMs are generally classified into two groups based on the different sets of assumptions and environments- continuous time models and discrete time models. In this paper, we analyze both the discrete as well as the continuous time models. Different types of discrete and continuous time models are compared, their assumptions and applications are studied.

Keywords- Software Reliability, SRGMs, Non Homogeneous Poisson Process, Calendar time, Execution time.

I. INTRODUCTION

Now a days, almost in every field, computers affect the people in one way or the other. Computers are used in various ways for many applications like that of air traffic control, nuclear reactors, aircraft, real-time sensor networks, industrial process control, and hospital health care affecting millions of people. Now as the computer system perform functions for almost all tasks and functions performed by computer systems are becoming essential and complicated and as the size and complexity of critical applications increases, the need to quantify and predict the reliability of computer system in various complex environments arises. [1]

Software Reliability is defined as the probability of failure free operation of software in a specified environment for a specified period of time [2]. With the increasing need of software with zero defects, predicting reliability of software systems is gaining more and more importance.

Various SRGM models are frequently used in the literature to estimate the reliability of a software product. A number of software reliability growth models have been developed under different sets of assumptions and environment.SRGMs can be classified into two categories. The first category comprises of the continuous time models. Continuous time models are those models which uses the execution time (i.e. CPU time) or calendar time. The second category comprises of the discrete time models. Discrete time models are those models which uses the test cases as a unit of fault removal period. Such models are called discrete time models, since the unit of software fault removal period is countable. [3]

Till Now, there are so many SRGMs have been developed that exist in the first category while only a few SRGM are developed that exist in the second category due to the problems and difficulties involved.

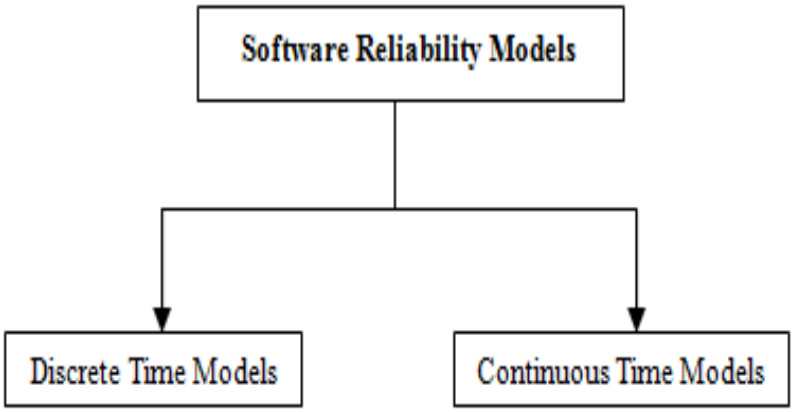

Fig 1. Types of Software Reliability Models

Software Reliability models are used to estimate the reliability of a software. We can classified the software reliability models in 3 parts. One is for continous time models, second is as discrete and the third as others category in which the other models can be grouped other than the continous and discrete models. But till now, more than 60% of the SRGMs are continous time while some are discrete models.

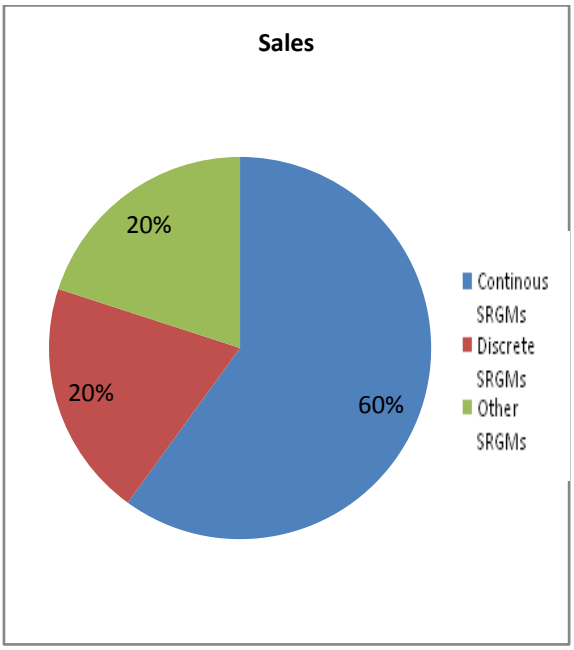

Fig 2. Classification of Software Reliability Models

II. CONTINUOUS TIME MODELS

1) Goel- Okumoto Model [1]

The Goel- Okumoto model is based on the following assumptions:_

a) From the failure detection point of view, faults in a program are mutually independent.

b) The probability is constant of the failures for faults actually detected. This means that the current number of faults in a program is proportional to the number of failures detected at any time.

c) Before further testing, the isolated faults are removed.

d) The software error which caused a failure is removed immediately and new errors are not introduced.

This is shown by the following equation:-

$$\frac{\partial m(t)}{\partial t} = b[a - m(t)]$$

Where

'a' is the expected total number of faults that exist in the software before testing .

and 'b' is the failure detection rate or the failure intensity of a fault.

The mean value function of the above equation be given as:-

$$m(t) = a\left(1 - e^{-bt}\right)$$

2) Yamada Delayed S-Shaped Model [1]

A stochastic model based on NHPP for a detected software error in which the growth curve is s-shaped for the observed failure data, of the number of detected software errors. This model can be characterized as a learning process of the testing team. The delayed S-shaped model is based on the following assumptions:-

a) From the failure detection point of view, faults in a program are mutually independent.

b) The probability is constant of the failures for faults actually detected.

c) The current number of faults in a program is proportional to the number of failures detected at any time.

d) The software's initial error content is a variable.

e) Errors present in the software system leads the system fail at random times.

f) The time between $(i-1)^{th}$ and i^{th} failures depends on the time to the $(i-1)^{th}$ failure.

g) The software error which caused a failure is removed immediately and new errors are not introduced.

This is shown by the following equation:-

$$b(t) = \frac{b^2 t}{bt + 1}$$

Where

'b' is the error detection rate per error in the steady-state.

The mean value function is given by

$$m(t) = a\left[1 - (1 + bt)e^{-bt}\right]$$

3) Inflection S-Shaped model [1]

This model is based on the dependency of faults. The inflection S-Shaped model is based on the following assumptions:-

a) Before the removal of some faults, some other faults are not detected.

b) The current number of faults in a software program is proportional to the number of failures is proportional to the number of failures detected at any time.

c) Each detectable fault failure rate is constant and identical.
d) Removal of isolated faults entirely.
Assume

$$b(t) = \frac{b}{1+\beta e^{-bt}}$$

Where
'b' represents the failure-detection rate and
'β' represents the inflection factor.
The mean value function is given by:-

$$m(t) = \frac{a}{1+\beta e^{-bt}}\left(1-e^{-bt}\right)$$

4) Musa's Basic Execution Time Model [4]
The Musa-Basic model, also termed as the exponential model, is given by the following mean value:-

$$m(t) = \beta_0^E\left(1-e^{-\beta_1^E t}\right)$$

Where β_0^E : is the expected number of failures and β_1^E is the hazard rate or in other words "the amount that each fault contributes to the overall failure rate".
5) Musa Okumoto logarithmic Poisson Model [4]
In software cost estimation models with high accuracy, this Musa Okumoto logarithmic model is used. This model is also known as the logarithmic model.
The mean value function of the model is given by:-

$$\mu(t) = \beta_0^L Ln\left(1+\beta_1^L t\right)$$

The required data to build this model are the one from the time between failures and the time of failure.

III. DISCRETE TIME MODELS

1) Discrete logistic curve models [8]
A logistic curve model is a deterministic model which has been applied to many SRGMs.It can be described as

$\frac{dL(t)}{dt} = \frac{\alpha}{K}L(t)(k-L(t))$

Where α and k are constant parameters which can be known only by regression analysis, and L(t) is the cumulative number of software failures
1.1) Discrete logistic curve model with Morishita's equation
Morishita's states the product of the cumulative number of faults detected up to discrete time (n+1) and number of remaining faults at discrete time n in the software is proportional to the number of faults detected.
Morishita's gives the following equation as a discrete equation as:-

$L_{n+1}-L_n = \delta\frac{\alpha}{k} L_{n+1}(k-L_n)$

It has an exact solution:-

$$L_n = \frac{k}{1+m(1-\delta\alpha)^{\frac{t_n}{\alpha}}}$$

Where k= total number of software failures,
m = constant of integration
1.2) Discrete logistic curve model with Hirota's equation
This model states that the number of faults detected during time difference is proportional to the product of the cumulative number of faults detected up to discrete time n and the number of remaining faults at discrete time n+1 in the software.
Hirota gives the following equation as a discrete equation

$L_{n+1}-L_n = \delta\frac{\alpha}{k}L_n(k-L_{n+1})$

This has an exact solution

$$L_n = \frac{k}{1+m\left(\frac{1}{1+\delta\alpha}\right)^{\frac{t_n}{k}}}$$

Where k= total number of software faults
m = constant of integration
2) Discrete Gompertz Curve Model [8]
This model is from S- shaped SRGMs class which gave good approximations to a cumulative number of software faults observed in testing software.
This model gives the following equation

$\frac{dG(t)}{dt}$ = (logb) G(t) $\log\frac{G(t)}{k}$

where G(t) is cumulative number of software faults detecting up to testing time,
and k = initial fault content

By integrating the above equation and assume G(0) = ka,
G(t) can be written as
$G(t) = ka^{b'}$
where k represents the initial fault content
An exact solution of this equation is

$$G_n = ka^{(1+\delta \log b)^n}$$

IV. DIFFERENT MODELS ALONG WITH THEIR ASSUMPTIONS AND APPLICATIONS

Model Name	**Proposed Year**	**Assumptions**	**Mean Value Function(m(t)/ Exact Solution(G(t))**	**Applications**
1) Goel -Okumoto	1979	a) From the failure detection point of view, faults in a program are mutually independent. b) The probability is constant of the failures for faults actually detected. This means that the current number of faults in a program is proportional to the number of failures detected at any time. c) Before further testing, the isolated faults are removed. d) The software error which caused a failure is removed immediately and new errors are not introduced.	$m(t) = a\left(1-e^{-bt}\right)$	This model can be used to predict the Mean Test Cases to Failures(MTCTF) and can also be used to estimate the Remaining Software Defect Estimation (RSDE) on actual software failure data.
2) Yamada Delayed S-Shaped Model	1984	a) From the failure detection point of view, faults in a program are mutually independent. b) The probability is constant of the failures for faults actually detected. c) The current number of faults in a program is proportional to the number of failures detected at any time. d) The software's initial error content is a variable. e) Errors present in the software system leads the system fail at random times. f) The time between $(i-1)^{th}$ and i^{th} failures depends on the time to the $(i-1)^{th}$ failure. g) The software error which caused a failure is removed immediately and new errors are not	$m(t) = a\left[1-\left(1+bt\right)e^{-bt}\right]$	This model is used in the situations in which the observed growth curve of the cumulative number of detection error is s-shaped. Yamada delayed s-shaped model can also be used to estimate the Remaining Software Defect Estimation (RSDE) on actual software failure data.

		introduced.		
3) Inflection S-Shaped model	1984	a) Before the removal of some faults, some other faults are not detected. b) The current number of faults in a software program is proportional to the number of failures is proportional to the number of failures detected at any time. c) Each detectable fault failure rate is constant and identical. d) Removal of isolated faults entirely.	$m(t) = \frac{a}{1+\beta e^{-bt}}\left(1-e^{-bt}\right)$	This model is best in situation where we need to find out the expected number of remaining errors at a specified time.
4) Musa's basic execution time Model	1985	a) From the failure detection point of view, faults in a program are mutually independent. b) The probability is constant of the failures for faults actually detected. This means that the current number of faults in a program is proportional to the number of failures detected at any time. c) Before further testing, the isolated faults are removed. d) The software error which caused a failure is removed immediately and new errors are not introduced.	$m(t) = \beta_0^E\left(1-e^{-\beta_1^E t}\right)$	This model is used in applications where the cost estimation is required before the release of the software.
5) Musa- Okumoto logarithmic poisson model		The execution time, i.e., the actual processing time used in executing the program is the best time domain for expressing reliability.	$m(t) = \beta_0^L L_n\left(1+\beta_1^L t\right)$	This model is used especially for the execution time data but it can also be applied to calendar time data by applying a conversion from calendar to execution time
6) Yamada- Osaki Exponential Growth Model	1985	The Failure intensity of faults within different modules are assumed to be different while the failure intensity of faults within the same module are assumed to be the same. Assume that the expected numbers of faults detected for each module are exponential.	$m(t) = a\sum_{i=1}^{k} p_i\left[1-e^{-b_i t}\right]$	We may use Yamada-Osaki Exponential Growth Model where the failure intensity of faults within different modules are assumed to be different while the failure intensity of faults within the same module are assumed to be the same and the expected number of faults detected are exponential.

7) Pham-Nordmann-Zhang(PNZ) Model	1999	a) The introduction rate is a linear function time-dependent overall fault content function. b) The fault detection rate function is non-decreasing time-dependent with an inflection S-shaped model.	$m(t)=\frac{a}{1+\beta e^{-bt}}\left(\left[1-e^{-bt}\right]\left[1-\frac{\alpha}{\beta}\right]+\alpha t\right)$	The PNZ model incorporates the imperfect debugging phenomenon by assuming that faults can be introduced during the debugging phase at a constant rate of fault per detected fault
8) Pham Exponential Imperfect Debugging Model	2000	a) The introduction rate is an exponential function of testing time. b) The error detection rate function is non-decreasing with an inflection S-shaped model.	$m(t)=\frac{\alpha b}{b+\beta}\left(\frac{e^{(\beta+b)t}-1}{e^{bt}+c}\right)$	This model can be used where the error detection rate is an exponential function of testing time.
9) Discrete Logistic Curve Models 9.1) Discrete Logistic Curve Model with Morishita's equation 9.2) Discrete logistic curve model with Hirota's equation		Morishita's states the product of the cumulative number of faults detected up to discrete time (n+1) and number of remaining faults at discrete time n in the software is proportional to the number of faults detected. The number of faults detected during time difference is proportional to the product of the cumulative number of faults detected up to discrete time n and the number of remaining faults at discrete time n+1 in the software.	$L_n=\frac{k}{1+m\left(1-\delta\alpha\right)^{\frac{t_n}{\alpha}}}$ $L_n=\frac{k}{1+m\left(\frac{1}{1+\delta\alpha}\right)^{\frac{t_n}{k}}}$	This model is deterministic, even though it can be applied to many software-reliability growth processes in software testing. This model is one of the best and simplest models for estimating an S-shaped software reliability growth curve model.
10)Discrete Gompertz Curve Model		Gave good approximations to a cumulative number of software faults observed in testing software.	$G_n=ka^{(1+\delta\log b)^n}$	Gompertz Model is from one of the S-shaped SRGM models. This model give good approximations to a cumulative number of software faults observed in testing software.

Table 1. Comparison of Different Software Reliability Models Based on Their Assumptions and Applications.

V. CONCLUSION

Reliability models are a powerful tool for predicting and assessing software reliability. In studying hardware and software reliability problems, NHPP models have been successfully used the SRGMs are especially used to describe processes of failure which possess certain trends, such as reliability growth, and hence making the application of NHPP models to software reliability analysis easily implemented.In this paper, we first categorize the various SRGMs into continous time models and discrete time models and then studied the various SRGMs under continous time models and discrete time models.

A release of a software product is always be a trade-off between early release and the product release deferral to enhance functionality. We conclude that the SRGMs can help the software designer to decide when the software system is ready for release, if the reliability of the software has reached a given threshold.

REFERENCES

[1] Hoang Pham, "System Software Reliability", Springer Series in Reliability Engineering, 2006.
[2] Chin-Yu Huang, Chu-Ti Lin, Chuan-Ching Sue, "Software Reliability Prediction and Analysis during Operational Use", IEEE, 2005.
[3] D. N. Goswami, Sunil K. Khatri, Reecha Kapur, "Discrete Software Reliability Growth Modeling for Errors of Different Severity Incorporating Change-point Concept", International Journal of Automation and Computing, October 2007.
[4] Sonia Meskini, Ali Bou Nassif, Luiz Fernando Capretz, "Reliability Models Applied to Mobile Applications", International Conference on Software Security and Reliability Companion, 2013.
[5] Saisuke Satoh, Shigeru Yamada, "Discrete Equations and Software Reliability Growth Models", IEEE, 2001.
[6] M. R. Lyu, "Handbook of Software Reliability Engineering", McGraw Hill, 1996.
[7] M. R. Lyu, "Handbook of Software Reliability Engineering", McGraw Hill, 1996.
[8] Y. K. Malaiya and J. Denton, "What do the software Reliability Growth Model Parameters Represents?", 8th IEEE International Symposium on Software Reliability Engineering, Albuquerque, 1996
[9] R. Lai and M. Garg, "A Detailed Study of NHPP Software Reliability Models", Journal of Software, 2012.
[10] J. D. Musa, A.lannino and K. Okumoto, "Software Reliability – Measurement, Prediction, Applications", McGraw Hill, 1987.
[11] W. Xia, L.F. Capretz and D. Ho, "A Neuro- Fuzzy Model for Function Point Calibration, WSEAS Transactions on Information Science and Applications", 2008.
[12] A.B. Nassif, L.F. Capretz and D. Ho, "Towards an Early Software Estimation Using Log-Linear Regression and a Multilayer Perception Mode', Journal of Systems and Software, 2013.
[13] A.B. Nassif, L.F. Capretz and D. Ho, "Software Estimation in the Early Stages of the Software Life Cycle", International Conference on Emerging Trends in Computer Science, Communication and Information Technology, Nanded, India, 2010.
[14] A.B. Nassif, L.F. Capretz and D. Ho, :Estimating Software Effort Based on Use Case Point Model Using Sugeno Fuzzy Inference System", 23rd IEEE International Conference on Tools with Artificial Intelligence(ICTAI), Boca Raton, FL, USA, 2007.
[15] P. K. Kapur, S. Younes, S. Agarwala, "Generalise Erlang Model with N Types of faults", ASOR Bulletin, 1995.
[16] P. K. Kapur, A.K. Bardhan, O. Shatnawi, "Why Software Reliability Growth Modeling Should Define Errors of Different Severity", Journal of the Indian Statistical Association, 2002.
[17] P. K. Kapur, V. B. Sing, S. Anand, V.S.S. Yadavalli, "Software Reliability Growth Model with Change Point and Effort Control Using a Power Function of the Testing Time", International Journal of Production Research, November, 2006.
[18] P. K. Kapur, A. Kumar , K. Yadav, S.Khatri, "Software Reliability Growth Modeling for Errors of Different Severity Using Change Point", International Journal of Reliability, Quality and Safety Engineering, 2007
[19] P. K. Kapur, O.Shatnawi, O. Singh, "Discrete Imperfect Software Reliability Growth Models under Imperfect Debugging Environment", In Proceedings of the International Conference on Multimedia and Design, N. J. Rajaram, A. K. Verma, Arena Multimedia & IIT-Mumbai, Mumbai, 2002.
[20] P. K. Kapur, R. B. Garg, S. Kumar, "Contributions to Hardware and Software Reliability", World Scientific, Singapore, 1999.
[21] S. Yamada, S. Osaki, "Discrete Software Reliability Growth Models", Journal of Applied Stochastic Models and Data Analysis, 1985.
[22] S. Yamada, "A Stochastic Software Reliability Growth Model with Gompertz Curve", Trans. IPS Japan, 1992.

