International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 2 Issue 8, August-2013, pp: (82-86), Available online at: www.erpublications.com

A study of software reliability models based on execution time

Deepak Juneja[1], Jitender Kumar[2]
[1]Assistant Prof., Geeta Engineering College, Haryana, India
[2]M. Tech. Scholar, Sri Sukhmani Institute of Engineering and Technology, Derabassi, Punjab, India

Abstract: The ability of software to produce its desired results in specific amount of time under certain conditions is known as software reliability. We review in this paper basic reliability terms, bath tub curve for hardware reliability and its corresponding software reliability curve. The paper also includes a brief classification of the reliability models by giving main emphasis on the reliability models based on execution time. Basic Execution Time Model and Logarithmic Poisson Execution Time Model. In conclusion there are summary of knowledge about the necessity of reliability models.

Keywords: Software reliability, failure, fault, basic execution time model, logarithmic Poisson execution time model.

I. INTRODUCTION

Software reliability is not only the most important quantitative indicator of the quality of software, and is user-oriented view. The definition of software reliability is willingly similar to that of hardware reliability in order to compare and assess the reliability systems composed of hardware and software components. Software reliability is a concept that defines the ability of software to produce its desired results in specific amount of time under certain conditions.

We need a number of definitions to introduce the concept of software reliability:

- **Errors:** these are human actions that result in the software containing a fault. Examples of such fault include deviation from user's requirements, coding error etc.
- **Failure:** It is something dynamic. It is the departure of the external results of program operation from requirements. These are observed during testing and operation.
- **Faults:** A fault is the defect in the program that, when executed under particular conditions, causes a failure.
- **Software Reliability :** It is defined as the ability of a system or component to perform its required functions under stated conditions for a specified period of time. It is the probability that the software will not cause the failure of a product for a specified time under specified conditions. This probability is the function of the inputs to and use of the product as well as the function of the existence of faults in the software.

II. REL1ABILITY CURVES

Over time, hardware exhibits the failure characteristics shown in Figure 1, known as the bathtub curve. Period A, B and C stands for burn-in phase, useful life phase and end-of-life phase.

Fig 1: Bathtub curve for hardware reliability

Software reliability, however, does not show the same characteristics similar as hardware. A possible curve is shown in Figure 2 if we projected software reliability on the same axes. There are two major differences between hardware and software curves. One difference is that in the last phase, software does not have an increasing failure rate as hardware does. In this phase, software is approaching obsolescence; there are no motivation for any upgrades or changes to the software. Therefore, the failure rate will not change. The second difference is that in the useful-life phase, software will experience a drastic increase in failure rate each time an upgrade is made. The failure rate levels off gradually, partly because of the defects are found and fixed after the upgrades.

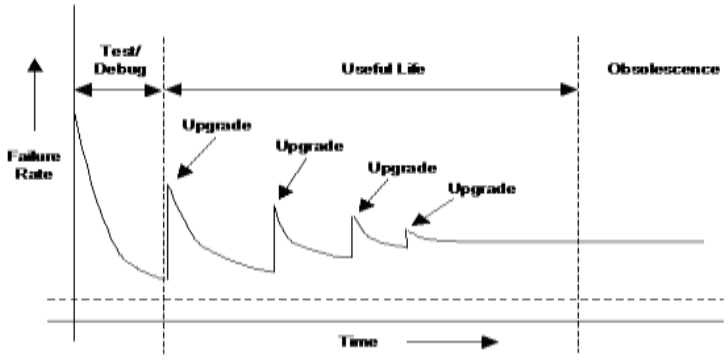

Fig 2: Software reliability curve

III. CLASSIFICATION OF SOFTWARE RELIABILITY MODELS

Many software reliability models (SRMs) have been developed. The two main categories among these are predictive models and assessment models.

A. Predictive models

Predictive models primarily address the reliability of the software early in the life cycle at the requirements or at the preliminary design level or even at the detailed design level in the waterfall life cycle process or in the first spiral of a spiral software development process. These models could be used to assess the risk of developing software under a given set of requirements and for specified personnel before the project truly starts.
The Air Force's Development Centre (RADC) proposed one of the first predictive models.

B. Assessment models

Most existing SRMs may be grouped into four categories:

1) ***Time between failure categories*** includes models that provide estimates of the times between failures. The key model in this category is Jelinski-Moranda model.

2) ***Failure count category*** is the number of faults or failure experienced in specific intervals of time. It includes modes like Musa – Okumoto's logarithmic Poisson time model, Shooman's exponential model etc.

3) ***Fault seeding category*** includes models to assess the number of faults in the program at time 0 via seeding of extraneous faults. Specific model in this group is Mills seeding model.

4) ***Input-domain based category*** includes models that assess the reliability of a program when the test cases are sampled randomly from well-known operational distribution of inputs program. Specific model in this group are nelson's model, Bastani's model etc.

IV. RELIABILITY MODELS BASED ON EXECUTION TIME

A. Basic Execution Time Model

The model was developed by J.D Musa in 1979 and is based on execution time. It is assumed that failure may occur according to a non homogenous Poisson Process (NHPP).

Real world events may be described using Poisson Processes. Examples of Poisson Processes are:

- Number of telephone calls expected in a given period of time.
- Expected number of road accidents in a given period of time.
- Expected number of persons visiting in a shopping mall in a given period of time.

In this case processes are non homogenous, since failure intensity changes as a function of time. In this model, the decrease in failure intensity, as a function of the number of failures observed, is constant and is given as:

$\lambda(\mu)=\lambda_o(1-\mu/V_o)$

Where λ_o : initial failure intensity at the start of execution.
V_o : number of failures experienced if program is executed for infinite time period.
μ : average or expected number of failures experienced in a given point in time.

The relationship between failure intensity (λ) and mean failures experienced (μ) is given in figure 3:

Fig 3: Failure intensity λ as a function of μ for basic model

The slope of the failure intensity can be obtained by finding its first derivative and is given as:

$$d\lambda/d\mu = -\lambda_o/V_o$$

This model implies a uniform operational profile if all input classes are selected equally often, the various faults have an equal probability of manifesting themselves. The correction of any of those faults then contributes an equal decrease in the failure intensity. The negative sign shows that there is a negative slope which means a decrementing trend in failure intensity.

The relationship between the execution time(τ) and mean failures experienced(μ) is shown in figure 4:

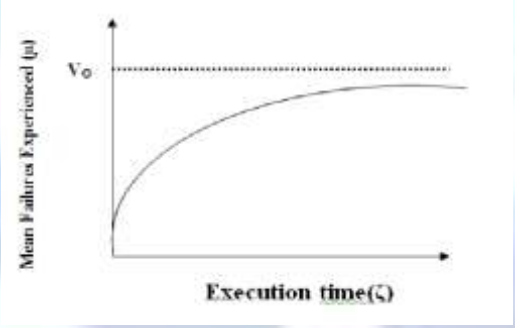

Fig 4: Relationship between τ & μ for basic model

For a derivation of this relationship equation 1 can be written as:

$d\mu(\tau)/d\tau = \lambda o\ (1-\mu(\tau)/Vo)$

The above equation can be solved for $\mu(\tau)$ and results in

$\mu(\tau)=V_o(1-\exp(-\lambda_o\tau/V_o))$

The failure intensity as a function of execution time is shown in figure 5:

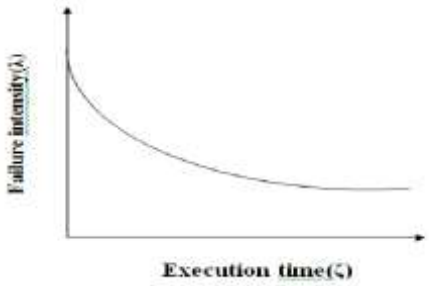

Fig 5: Failure intensity versus execution time for basic model

The relationship is useful for determining the present failure intensity at any given value of execution time.

B. Logarithmic execution time model

This model is also developed by Musa. The failure intensity function is different here as compared to basic model. In this case, failure intensity function (decrement per failure) decreases exponentially whereas it is constant for basic model.
The failure intensity function is given as:
$\lambda(\mu)= \lambda_o \exp(-\theta\mu)$
Where θ: is called the failure intensity decay parameter.

The relationship between the failure intensity (λ) and mean failures experienced (μ) is shown in figure 6:

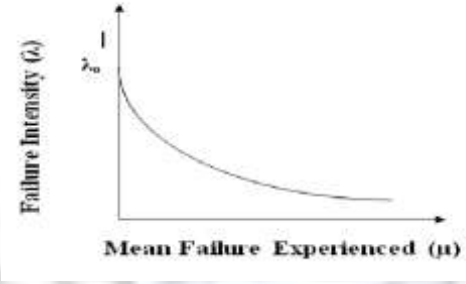

Fig 6: Relationship between μ & λ

The 'θ' represents the relative change of failure intensity per failure experienced . The slope of failure intensity function is:

$$d\lambda/d\mu= -\lambda o\theta \exp(-\mu\theta)$$
$$d\lambda/d\mu = -\theta\lambda$$

The relationship between execution time and mean failures experienced is given in figure 7.

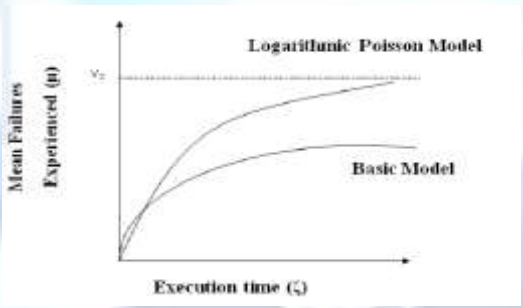

Fig 7: Relationship between τ & μ

The expected number of failures for this model is always infinite at infinite time. The relation for number of failures is given by:

$$\mu(\tau)= 1/\theta \text{ Ln } (\lambda_o\theta\tau+1)$$

The failure intensity of the Logarithmic Poisson execution Time model drops more rapidly initially than that of the basic model and later it drops more slowly.
The expression for failure intensity is given as:

$\lambda(\tau)=\lambda_o/(\lambda_o\theta\tau+1)$

The relations for the additional number of failures and additional execution time in this model are:

$\delta\mu=1/\theta Ln(\lambda p/\lambda F)$

and $\delta\tau=1/\theta(1/\lambda F-1/\lambda p)$

Where λp: present failure intensity.
λF: failure intensity objectives.
Hence, at larger values of execution time, the logarithmic Poisson model will have larger values of failure intensity than the basic model.

V. CONCLUSION

Software becomes more reliable over time, instead of wearing out. It either works in a given environment or it does not. Hardware reliability curve is a bath tub curve which shows that failure rate firs decreases and then increases during the last phase i.e. wear out phase while the software reliability curve shows that the software retires only if it becomes obsolete. Software reliability models are used to access the reliability of particular software.

In basic execution time model, processes are non homogenous, since failure intensity changes as a function of time. In this model, the decrease in failure intensity, as a function of the number of failures observed, is constant.
In logarithmic execution time model, failure decreases exponentially.

REFERENCES

[1]. Goel, A., L. Software reliability models: Assumptions, limitations, and applicability, In IEEE Trans. Soft. Eng, SE11, 1985.
[2]. Musa, J., D. Operational profiles in software reliability engineering, In IEEE Software, 1993.
[3]. H. Pham, System Software Reliability. : Springer London, 2006.
[4]. J.D. Musa and K. Okumoto, "A logarithmic poisson executiontimemodel for software reliability measurement", 7th Int'l Conference on Software Engineering (ICSE), 1984.
[5]. Software Reliability Modeling Programs. Reliability and Statistical Consultants Ltd [R].Version 1.0, 1988.
[6]. Lyu M. R., and Nikora A.P., CASRE - A Computer-Aided Software Reliability Estimation Tool[C]. CASE 92 Proceedings, Montreal, Canada, 1992: 264-275.
[7]. Ren Zuo, Xu.(ED.), Software Reliability Expert System (SRES). Beijing, CN: Tsinghua university press, 1996.