Volume 4, Issue 11, November 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Software Reliability with Random Test Cases

Pankaj Dalal
Research Scholar Mewar University,
India

Dr. D. S. Rao
Director, IST, INDORE, MP,
India

Abstract: ***The prime concern in Software quality is Reliability. There are many approaches to calculate reliability. Here we are calculating reliability by using Mean Time to Failure (MTTF). In this paper we show that reliability graph is non linear curve. The selection of test cases are made randomly. The 12 week data is used to calculate MTTF.***

Keyword: Software Engineering, Software Reliability, Reliability Metrics

I. INTRODUCTION

The probability of failure-free operation of a computer program in a specified environment for a specified time, called reliability.

Software Reliability is dynamic in nature. It depends upon the problem occurrence, the cost and time on solution. The probability of failure-free operation in a specified environment over a specified time is shown below.

The expected curve for software is shown in Figure 1. [2]

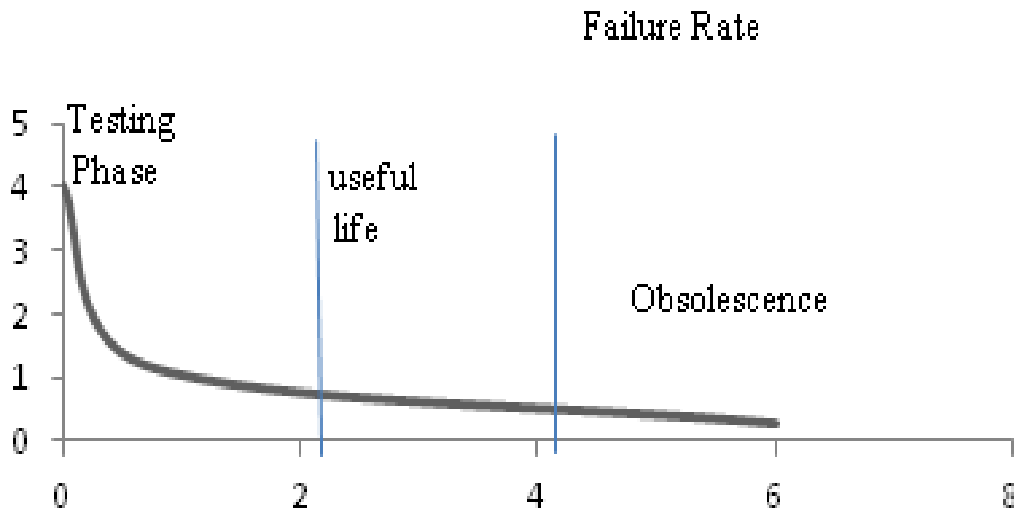

Fig: 1 Software Reliability Curve (Failure Rate Versus Time)[2]

II. MEASUREMENT OF SOFTWARE RELIABILITY

Software reliability for any system can be calculated in term of software reliability metrics.

A. Reliability Metrics

Reliability metrics are quantitative measure of a software product. The quantifying reliability metrics are:[1][2][6]

1. MTTF (Mean Time to Failure).

The products have an average lifetime and calculated in terms of Mean Time to Failure (MTTF). The mean time for which expected to be operational. It is the average time/trisection between observed system failures.

2. MTTR (Mean Time to Repair)

The time required to fix the error on failure. MTTR measures the average time that it takes to track the errors that leads to failure and then to fix them.

The MTTF is the mean time for which a component is expected to be operational. The MTTF is the average time between observed system failures. An MTTF of 500 i.e one failure in every 500 time units. The time units are totally dependent on the system and it can even be specified in the number of transactions. MTTF is relevant for systems with long transactions, where the system processing takes a long time. It should be longer than transaction length. [2][6]

3. MTBF (Mean Time Between Failures)

The addition of the MTTF and MTTR metrics is MTBF metric:

$$MTBF = MTTF + MTTR$$

Thus, a MTBF are real time and not the execution time as in MTTF. Thus, a MTBF of 300 hours indicates that the next failure is ***expected to occur only after 300 hours. [2][6]***

4. POFOD (Probability of Failure on Demand)

POFOD is the likelihood that the system will fail when a service request is made. If one failure out of a thousand service requests the POFOD is 1/1000. It is an important measure for safety critical systems and kept as low as possible. [2][6]

5. ROCOF (Rate of Occurrences of Failure)

ROCOF is the frequency of occurrence with which unexpected behavior is likely to occur. A ROCOF/failure intensity of 2/100, two failures are likely to occur in each 100 operational time units. [2][6]

6. AVAIL (Availability)

Availability is the probability of system available for use at a given time. The Availability is 0.998; in every 1000 time units the system would be available for 998 of these. [2][6]

III. SOFTWARE RELIABILITY

The Software reliability is calculated in term of MTTF. It is the derivative of failure intensity. The Failure intensity is number of failure per day and the MTTF is reverse of failure intensity.

IV. THE METHODOLOGY

Software reliability calculation has the following steps:

- Fault Data Collection
- Analysis of Fault Data
- Calculation with appropriate method

A. Data Collection

In our work we have collected live defect data from the Cimcon Pvt. Limited to calculate software reliability.

B. Data Analysis

Collected defect data is in standard format so it has converted in our required format to calculate software reliability which have number of failures and test time.

C. Reliability Calculation

Software reliability is calculated in terms of software metrics as discussed in section 2. In this research paper we compute actual software reliability calculated in term of MTTF.

MTTF is the derivative of failure intensity. Failure intensity is number of failure per day. And the MTTF is reverse of failure intensity.

The failure intensity and MTTF is calculated as in equation eq.1. & eq. 2. [3][7][8]

$$\text{Failure intensity(i)} = \frac{\text{No. of Failure}}{\text{Time of failure}} \quad \text{eq. 1.}$$

$$\text{MTTF} = \frac{1}{\text{Failure Intensity}} \quad \text{eq. 2.}$$

V. DATA ANALYSIS

The appropriate data points in week are created with the collected data set for computation shown in table 1.

Table 1 defect data analysis

Sr. No.	No. of Week	No. of Defect	cumulative error	Time
1	Week 1	14	14	7
2	Week 2	10	24	14
3	Week 3	6	30	21
4	Week 4	0	30	28
5	Week 5	0	30	35
6	Week 6	0	30	42
7	Week 7	0	30	49
8	Week 8	0	30	56
9	Week 9	3	33	63
10	Week 10	0	33	70
11	Week 11	0	33	77
12	Week 12	4	37	84

VI. RELIABILITY CALCULATIONS

Here we calculate actual reliability and tabulated in table 2.

Table 2 actual software reliability

Sr. No.	No. of Week	intensity	MTTF
1	Week 1	2	0.5
2	Week 2	1.714	0.58
3	Week 3	1.428	0.7
4	Week 4	1.071	0.93
5	Week 5	0.857	1.16
6	Week 6	0.714	1.4
7	Week 7	0.612	1.63
8	Week 8	0.535	1.86

9	Week 9	0.523	1.9
10	Week 10	0.471	2.12
11	Week 11	0.428	2.33
12	Week 12	0.44	2.27

The actual software reliability in table 2 is calculated from table 1.

VII. RESULTS

A graph for MTTF values is plotted in fig 2 using data set of table 2.

The graph shows that Reliability is non linear hence the estimations of accurate prediction is not possible. The accurate software reliability prediction is not possible.

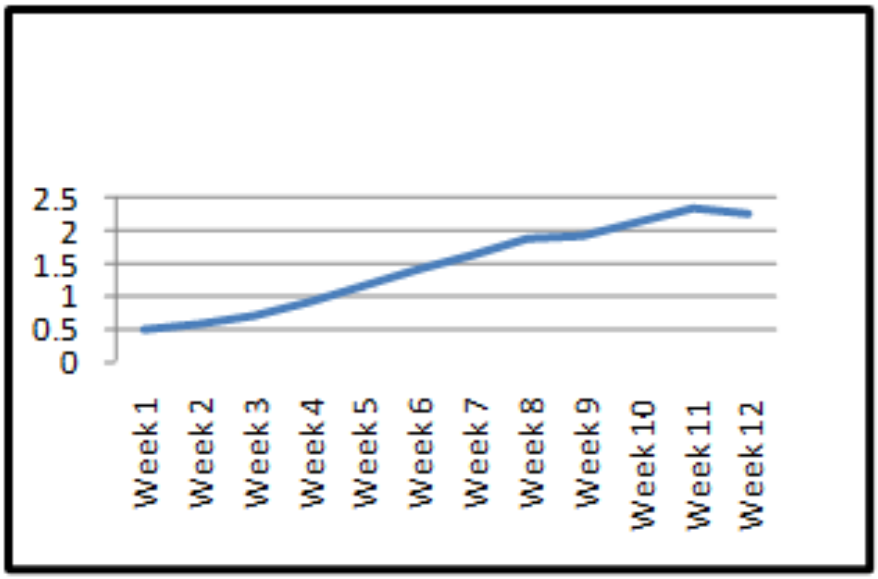

Fig. 2 Software reliability graph

VIII. CONCLUSION

The actual software reliability graph concludes non linear behavior which could not used to estimate accurate software reliability. The various software reliability growth models are used to smoothen the curve. The fault deduction is test case dependent. Here test cases are randomly selected.

The Test case selection must have plan with criteria of selection, which will improve the reliability. The Test case selection during debugging must be plan in design phase itself. The partial test cases can be applied during debugging, this will reduce testing time & cost. The module wise test cases are designed with the help of specification and Software operational Profile. The testing if starts from debugging phase, it will certainly improve the reliability curve and the reliability.

ACKNOWLEDGEMENT

We acknowledgement the support of live defect data from the Cimcon Pvt Limited Ahmedabad.

REFERENCES

[1] J. D. Musa, A. Iannino, and K. Okumoto, Software Reliability, Measurement, Prediction, Application. New York: McGraw-Hill, 1987.

[2] Amit Kumar Sharma, MTech(IT)1, Maneesha Sharma 2 , Balesh Kaushik 3" Software Reliability Engineering & Its Challenges" Proceedings of National Conference on Challenges & Opportunities in Information Technology (COIT-2007) RIMT-IET, Mandi Gobindgarh. March 23, 2007

[3] Pankaj Nagar and Blessy Thankachan "Software Reliability Engineering–A Review" International Journal of Applied Physics and Mathematics, Vol. 1, No. 2, September 2011

[4] Apoorva Singhal1, Ankur Singhal23 "A Systematic Review of Software Reliability Studies"

[5] Scott Dick, Member, IEEE, Cindy L. Bethel, and Abraham Kandel, Fellow, IEEE," Software-Reliability Modeling: The Case for Deterministic Behavior"

[6] Pankaj Jalote, Brendan Murphy, Mario Garzia, Ben Errez," Measuring Reliability of Software Products"

[7] Bev Littlewood and Lorenzo Strigini," Software Reliability and Dependability: a Roadmap", Proc. ICSE 2000,22nd International conference on Software Engineering.

[8] "IEEE Recommended Practice on Software Reliability" Sponsor Standards Committee of the IEEE Reliability Society Approved 27 March 2008 IEEE-SA Standards Board

