# Software Reliability Models: Failure rate estimation

Animesh Kumar Rai
*M.Tech Student, Department of information Technology*
*Amity School of Engineering and Technology*
*Amity University, Noida, Uttar Pradesh*

Rana Majumdar
*Assistant Professor, Department of Information Technology*
*Amity school of Engineering and Technology*
*Amity University, Noida, Uttar Pradesh*

**Abstract- Software reliability is one of the important factors which decide the quality of the software .Software is thoroughly checked and errors are removed before it is delivered to the client. A key factor in the success of a software project is achieving the best-possible software reliability. Software reliability is a mathematical model which ensures that software development has been done within cost and time and it will not cause failure under specified conditions. Different types of SRMs are used for different phases of the software development life-cycle. In this paper we will compare failure rate of different software reliability models by using SPSS tool. Non-Homogeneous Poisson Processes (NHPPs) are the most commonly used models in the reliability analysis of software.**

**Index Terms— SRGMs, Software Reliability, NHPP, Failure rate.**

## I. INTRODUTION

The basic goal of software development is to produce high quality software at low cost and within time. As the size and complexity of the software grows the issues with the reliability of the software also grows. So there is requirement of a software reliability model which ensures that the operation is failure free. It is very desirable to know the probability of software failure or the rate at which software errors will occur. SRGMs are mathematical models which describes how software attains reliability when the faults are detected and removed[1]. It indicates when software is ready to release and it has gain the expected reliability level[7]. Many SRGMs have been proposed in the past to estimate the expected number of total defects (or failures) or the expected number of remaining defects/ failures. Some well known SRGMs are:-

1) Goel model (1985)
2) Goel and Okumoto model (1979)
3) Schick and Wolverton model.
4) Musa and Ackerman model (1989)
5) Jelinski and Moranda.

Although these models have some limitations. None of the models are good for all data sets[6]; they are ideal for some particular data set but unfit for other data set. In this paper we will consider some SRGMs and estimate the software failure rate of the models by using a common data set for each model and we will use SPSS tool to estimate the failure rate of the SRGMs and compare the results.

## II. LITERATURE REVIEW

### *2.1 Description of Different SRGMs*

The SRGMs are used in estimating the reliability metrics of software products. The following assumptions are made for software reliability modelling[3]:-

(i) The fault removal process follows the Non- Homogeneous Poisson process (NHPP)
(ii) The software system is subjected to failure at random time caused by faults remaining in the system.
(iii) The mean time number of faults detected in the time interval $(t, t+\Delta t)$ by the current test effort is proportional for the mean number of remaining faults in the system.
(iv) The proportionality is constant over the time.

(vi) Each time a failure occurs, the fault that caused it is immediately removed and no new faults are introduced.

Let’s have a view on some basic parameters before we proceed to details of software reliability models.
Let $\{Z(t), t \geq 0\}$ denote a counting process representing the cumulative number of faults detected by the time t. An SRGM based on an NHPP with the mean value function (MVF), *m*(t) can be formulated as[4]:
$P\{z(t)=n\} = e^{[m(t)]} * [m(t)]/n!$
Where n = 0, 1, 2, 3….. And m (t) represents the expected cumulative number of faults detected by the time t. The MVF *m* (t) is non-decreasing with respect to testing time t under the bounded condition m (∞) = *a*, where *a* is the expected total number of faults to be eventually detected. Knowing its value can help us to determine whether the software is ready to be released to the customers and how much more testing resources are required[6].
The failure intensity function at testing time t is:-
$\lambda(t) = d\,m(t)/dt = m'(t)$
The software reliability, i.e., the probability that no failures occur in (s, s+t) given that the last failure occurred at testing time s where ($s \geq 0$, $t>0$), is:-
$R(t|s) = \exp[-m(t+s) - m(t)]$
The fault detection rate per fault at testing time t is given by:-
$d(t) = m'(t)/(a-m(t)) = \lambda(t)/(a-m(t))$
In next section, we will discuss the different software reliability models.

2.2 Goel and Okumoto Model.
This model assumes that [1](1) the number of failure in non overlapping intervals are independent and (2) the expected number of failure is proportional to the expected number of undetected errors.
Here, the fault rate with hypothesized mean value function is:-

$m(\infty) = a*(1-e^{(-bt)})$

where m (∞) = expected number of faults detected eventually.
b = fault detection rate.
a = expected total number of faults.

2.3 Jelinski and Moranda Model
This model assumes that:-
1) Failure data take the form of successive independent times between failures.
2) A constant failure rate between failures.
3) A failure rate is proportional to the software’s current fault content.

Here, the software failure rate is:-

$Z(ti) = \Phi[N-(i-1)]$

Where, ti = Time between the $(i-1)^{th}$ and $i^{th}$ failure.
N=number of initial errors in the program.
Φ=proportionality constant.

2.4 Schick and Wolvertan Model
Schick and Wovertan modified the J-M model by considering a time dependent failure intensity function and the time between failures to follow wiebull distribution[2].
Here, the software failure rate is:-

$\lambda(ti) = \Phi[N-(i-1)]ti$

where , ti = Time between the $(i-1)^{th}$ and $i^{th}$ failure.
N=number of initial errors in the program.
Φ=proportionality constant.

## III. WORKING AND METHODOLOGY

Here we will take a common data set for all the models that consists the time and cumulative faults. And using SPSS tool we will estimate the failure intensity of the models by applying the expressions for failure rate estimation. SPSS tool is IBM's Statistical tool for estimation and calculation. We also find the predicate values for these cumulative faults.

*3.1 Goel and Okumoto Model:-*
The fault rate with hypothesized mean value function is:-

$m(\infty) = a*(1-e^{-bt})$

where $m(\infty)$ = expected number of faults detected eventually.
b = fault detection rate.
a = expected total number of faults.

**Nonlinear Regression Analysis**

**Parameter Estimates**

| Parameter | Estimate | Std. Error | 95% Confidence Interval | |
|---|---|---|---|---|
| | | | Lower Bound | Upper Bound |
| A | 760.534 | 139.892 | 465.388 | 1055.680 |
| B | .032 | .007 | .016 | .048 |

**Correlations of Parameter Estimates**

| | a | b |
|---|---|---|
| A | 1.000 | -.998 |
| B | -.998 | 1.000 |

**ANOVA[a]**

| Source | Sum of Squares | df | Mean Squares |
|---|---|---|---|
| Regression | 953052.512 | 2 | 476526.256 |
| Residual | 2656.488 | 17 | 156.264 |
| Uncorrected Total | 955709.000 | 19 | |
| Corrected Total | 196108.947 | 18 | |

Dependent variable: m[a]

a. R squared = 1 - (Residual Sum of Squares) / (Corrected Sum of Squares) = .986.

And we got the predicate value for the data entered

| | | | | | | |
|---|---|---|---|---|---|---|
| 24.15 | | | | | | |
| 47.53 | | | | | | |
| 70.17 | | | | | | |
| 92.09 | | | | | | |
| 113.32 | | | | | | |
| 133.87 | | | | | | |
| 153.77 | | | | | | |
| 173.04 | | | | | | |
| 191.69 | | | | | | |
| 209.76 | | | | | | |

*3.2 Jelinski and Moranda*

the failure intensity decreases in a geometric progression on the occurrence of each individual failure,

λi = λ.K^i-1

Here, the software failure rate is:-

Z(ti) = Φ[N-(i-1)]

Where, ti = Time between the $(i-1)^{th}$ and $i^{th}$ failure.
N=number of initial errors in the program.
Φ=proportionality constant.

**Parameter Estimates**

| Parameter | Estimate | Std. Error | 95% Confidence Interval | |
|---|---|---|---|---|
| | | | Lower Bound | Upper Bound |
| A | -2.240 | 6641649523364408.000 | -14079668020150300.000 | 14079668020150290.000 |
| N | 39.459 | 316556122644877700.000 | -671069001895798910.000 | 671069001895799040.000 |
| I | 129.735 | 123612134975670992.000 | -262046019982973504.000 | 262046019982973792.000 |

**Correlations of Parameter Estimates**

| | a | N | I |
|---|---|---|---|
| A | 1.000 | -.925 | -.227 |
| N | -.925 | 1.000 | .580 |
| I | -.227 | .580 | 1.000 |

**ANOVA[a]**

| Source | Sum of Squares | df | Mean Squares |
|---|---|---|---|
| Regression | 759600.053 | 3 | 253200.018 |
| Residual | 196108.947 | 16 | 12256.809 |
| Uncorrected Total | 955709.000 | 19 | |
| Corrected Total | 196108.947 | 18 | |

Dependent variable: m[a]

a. R squared = 1 - (Residual Sum of Squares) / (Corrected Sum of Squares) = .954

*3.3 Schick and Wolvertan Model*

Assume the failure rate at the ith time interval increases withtime since the last debugging. The program failure rate function between the (i-1)[th] and the i[th] failure rate is:-

$$\lambda(t_i) = \phi[N-(i-1)]t_i$$

**Parameter Estimates**

| Parameter | Estimate | Std. Error | 95% Confidence Interval Lower Bound | 95% Confidence Interval Upper Bound |
|---|---|---|---|---|
| A | -1.595 | 9882116.022 | -20949151.717 | 20949148.527 |
| N | -76.502 | 239382896.025 | -507469146.327 | 507468993.324 |
| I | -63.208 | 209622394.459 | -444379688.056 | 444379561.641 |

**Correlations of Parameter Estimates**

| | a | N | I |
|---|---|---|---|
| A | 1.000 | -.526 | -.237 |
| N | -.526 | 1.000 | .951 |
| I | -.237 | .951 | 1.000 |

**ANOVA[a]**

| Source | Sum of Squares | df | Mean Squares |
|---|---|---|---|
| Regression | 949659.362 | 3 | 316553.121 |
| Residual | 6049.638 | 16 | 378.102 |
| Uncorrected Total | 955709.000 | 19 | |
| Corrected Total | 196108.947 | 18 | |

Dependent variable: m[a]

a. R squared = 1 - (Residual Sum of Squares) / (Corrected Sum of Squares) = .969.

## V. CONCLUSION

After analysing different types of software reliability models and calculating failure rate of the software product we analyzed that the software reliability models ensure the reliability of the software products as the failure rate is nearly 1 for the software models. As much as they are near to 1 they ensure the more reliability of the software product. The work is done using SPSS tool.

## VI. FUTURE WORK

The next goal is to optimize the reliability of the soft ware reliability models using genetic algorithms. Genetic programming can help to ensure the reliability of the software product is much better way,

REFERENCES

[1] "Software reliability Models: assumptions, limitations and applicability."IEEE transaction on software engineering volume SE 11,November 12,1985

[2] " Failure Correlation in Software Reliability Models" Katerina Goˇseva-Popstojanova, *Member, IEEE,* and Kishor S. Trivedi, *Fellow, IEEE*

[3] "Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value" *I.J. Information Technology and Computer Science,* 2013, 02, 1-14

[4] "Software Reliability Engineering–A Review" Pankaj Nagar and Blessy Thankachan. *InternationaL JournaL of Applied Physics and Mathematics, Vol. 1, No. 2, September 2011*

[5] "Software Reliability Growth Model with Logistic-Exponential Testing-Effort Function and Analysis of Software Release Policy" ACEEE Int. J. on Network Security , Vol. 02, No. 02, Apr 2011

[6] " Software Reliability Growth Modeling with New Modified Weibull Testing–effort and Optimal Release Policy" *International Journal of Computer Applications (0975 – 8887)Volume 6– No.12, September 2010*

[7] " SRGM with logistic-exponential Testing-effort function with change-point and Analysis of Optimal release policies based on increasing the test efficiency." Shaik Mohammad Rafi et al. / (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 03, 2010, 504-516