A New User Friendly and Efficient Tool for SRGM

[1]R.Subburaj and [2]B.Shanmugam.
[1]Professor and Consultant, [2]M.Tech Student
Department of Information Technology, SRM University – Chennai-603203, India
[1]subburaj.r@ktr.srmuniv.ac.in, [2]chammu.b@gmail.com

***Abstract*—Reliability of Software System can be enhanced through Reliability Growth Testing (RGT) process. Considering the impact of the reliability on success or failure of a software product, it is essential to model RGT process using Software Reliability Growth Models (SRGM). A large number of SRGMs have been proposed by a number of researchers. However the availability of software tools for this purpose is limited. Furthermore four flexible SRGMs have been proposed by Subburaj et al. which seem to fit the software failure data adequately irrespective of wide fluctuations in time between failures, presence or absence of the learning phenomenon of the testing team and Quality of Debugging. A Software Tool that facilitates estimation of parameters of the 4 model has been developed by the authors. The new SRGM tool developed has Graphical User Interface (GUI) and built with JAVA and R Language. This dedicated Open Source Tool (OST) is found to be equally accurate as that of general purpose tool such as MATLAB. This tool will help software reliability professionals to apply the SRGMs in their project easily and get accurate results.**

***Index Terms*—goodness of fit, parameter, RGT, R^2 metric, SRGM**

I. INTRODUCTION

Software reliability is a function of the number of failures experienced by a particular user of that software. A software failure occurs when the software is executing. It is a situation in which the software does not deliver the service expected by the user. Software failures are not the same as software faults although these terms are often used interchangeably. Errors committed by the software developer becomes a fault in the software which results in a failure when executed with certain combination of inputs and the software in a particular state.

Why Software Reliability is important:

Reliable software must include extra code to perform the necessary checking for exceptional conditions. This may have an effect on the program execution speed and increase the amount of storage required by the program. Nevertheless, Reliability should always take precedence over efficiency for the following reasons:

- Unreliable Software is liable to be discarded by users: If a company attains a reputation for unreliability because of single unreliable product, it is likely to affect future sales of all of that company's products.
- System failure costs may be enormous: for some applications, such as reactor control system or an aircraft navigation system software failure may lead to hazards such as loss of life, loss of mission etc.,
- It is usually possible to tune an inefficient system because most execution time is spent in small program sections. However an unreliable system is more difficult to improve as unreliability may be distributed throughout the system and causes of unreliability are difficult to find out.
- Inefficiency is predictable: unreliability usually surprises the user. Software that is unreliable can have hidden errors which can violate system and user data without warning and whose consequences are not immediately obvious, for example a fault in CAD program used to design aircraft might not be discovered until several plane crashes occur.
- Unreliable system may cause information loss: Since information is very expensive to collect and maintain, it may sometimes be worth more than the computer system on which it is processed. Hence reliability of the software system is quite important.

SRGMs

One of the objectives of system testing, which is the final testing phase in the Software Development Life Cycle (SDLC) is to correct as many faults as possible without introducing new faults in the process. The fault correction leads to growth of reliability. Therefore the system testing is also known as Reliability Growth Testing (RGT). The reliability models, used to plan and execute RGT in the SDLC are popularly known as Software Reliability Growth Models (SRGMs) since as a result of testing, faults are corrected and thereby reliability grows. SRGMs are found to be immensely useful to estimate quantum of reliability improvement during the testing phase of the Software Development Life Cycle. The SRGMs use the past failure data to estimate model parameters and facilitate deriving of various statistical inferences and in particular, about the current and future reliability of the software system. Quantitative information on the quality of the development process, testing process debugging process and also a clear criteria for exit from the testing phase can all be derived by using appropriate SRGMs during the software system testing phase of each project

We will review some of the features of popular flexible SRGMs in section II. We will discuss the necessity for software tool for an estimating model parameters, review some of the existing tools and give an overview of the SRGM tool developed by the authors in section III. Summary and Conclusions are given in Section IV.

II. FLEXIBLE SRGMS

After the publication of the popular Jelinski –Moranda [1] SRGM in the year 1972, a lot of researchers were attracted towards this area of research and a large collection of SRGMs were proposed. However, it may be noted that according to Brocklehurst et al. [2], 'in spite of much research effort, there is no universally applicable SRGM, which can be trusted to give accurate predictions of reliability in all circumstances'. This is due to many reasons. Some of the reasons why SRGMs do not describe data from a software projects are given below:

- Wide fluctuations and time between failures
- Some projects witness the learning phenomenon of the testing team and some don't. This results in S shaped growth and exponential growth of mean value function respectively.
- In Some projects most of the failures are observed in the early part of the testing phase.
- In some other projects most failures occur towards the end.
- Some models like Musa [3] basic execution time model published in the year 1972 and Goel-okumoto [4] Non Homogenous Poisson Process Model, published in the year 1979 assume exponential growth while Yamada et al [5]. 1985 assume S shaped growth of mean value function.

Because of the above reasons flexible models where necessitated. Goel [6] proposed a generalized non Homogenous Poisson Process model in the year 1985. Subburaj et al. [7] [8] [9] [10] in the late 2000s proposed four flexible SRGMs to describe the complex software failure data from current and future projects.

Subburaj-Gopal [7] [8] proposed two failure based SRGMs. These models assume the following for the sake of simplicity:

- debugging is perfect and therefore one fault leads to one failure

The mean value function and failure intensity function of Subburaj-Gopal [7] generalized NHPP model with Generalized Exponential function Rate Of OCcurrence Of Failures (ROCOF) are given below:

$$\mu(t) = N\left[1 - \exp\left(-\frac{t}{\theta}\right)\right]^{\beta} \qquad (1)$$

Where t is the time >0, θ is the scale parameter >0 and β is the shape parameter >0, N the eventual number of failures that will be observed over an infinite amount of testing time.

$$\lambda(t) = N\left(\frac{\beta}{\theta}\right)\exp\left(-\frac{t}{\theta}\right)\left[1 - \exp\left(-\frac{t}{\theta}\right)\right]^{\beta-1} \qquad (2)$$

λ(t) is the rate of change of mean value function over time or the number of failures per unit time and hence it is the derivative of mean value function given in Eq.(1). The hazard rate of the model increases monotonically if β>1, is constant if β=1 and decreases monotonically if β<1. In fact, it is this property which provides the flexibility to describe both monotonically decreasing and monotone failure rates observed in the software testing process.

The time between failures of the above model follows Generalized Exponential Distribution. Since NHPP models with Weibull function ROCOF may fit some of the failure datasets, Subburaj-Gopal [8] proposed another failure based model with shifted Weibull function ROCOF.

The mean value function and failure intensity function of Subburaj-Gopal [8] generalized exponential function ROCOF are given below

$$\mu(t) = N\left[1 - \exp\left(-\left(\frac{t-\gamma}{\theta}\right)^{\beta}\right)\right] \qquad (3)$$

Where N is the total number of failures expected to occur at infinite time. The failure intensity function of the proposed model is given by:

$$\lambda(t) = N\left(\frac{\beta}{\theta}\right)\left(\frac{t-\gamma}{\theta}\right)^{\beta-1}\left[\exp\left(-\left(\frac{t-\gamma}{\theta}\right)^{\beta}\right)\right] \qquad (4)$$

The hazard rate of the distribution increases monotonically if β>1, is constant if β=1 and decreases monotonically if β<1. In fact, it is this property and a flexible location parameter which enable the proposed model to best fit all patterns of variations of the failure intensity function.

The software development organizations need to collect quality metrics in order to manage the current project efficiently and continuously improve the quality of future projects. Hence Subburaj-Gopal-Kapur (SGK) [9] [10] proposed 2 fault based flexible SRGMs

The mean value function and failure intensity function of SGK [9] model are given below:

$$\mu(t) = \frac{a}{c}\left[1 - \exp\left(-\frac{ct}{\theta}\right)\right]^{\beta} \qquad (5)$$

Where a is the eventual number of faults that will be detected over an infinite amount of testing time, θ is the scale parameter >0, β is the shape parameter >0, and c is debugging index >0.
The failure intensity function λ(t) is the rate of change of mean value function over time, or the number of failures per unit time, and so it is the derivative of man value function with respect to time is given by:

$$\lambda(t) = a\left(\frac{\beta}{\theta}\right)\exp\left(-\frac{ct}{\theta}\right)\left[1 - \exp\left(-\frac{ct}{\theta}\right)\right]^{\beta-1} \qquad (6)$$

The hazard rate of the above increases monotonically if β >1, it is constant if β = 1, and it decreases monotonically if β<1. In fact, it is the phenomenon that enables us to propose a suitable NHPP model that brings out all the patterns of variations of failure intensity function correctly. Thus, the model address the learning phenomenon with β>1.

The mean value function and failure intensity function of SGK [10] model are given below:

$$\mu(t) = \frac{a}{c}\left[1 - \exp\left(-c\left(\frac{(t-\gamma)}{\theta}\right)^{\beta}\right)\right] \qquad (7)$$

where, a: the eventual number of faults that will be detected over an infinite amount of time
θ: scale parameter
β: shape parameter
γ: location parameter
c: debugging index

The shape parameter β provides an indication of overall testing efficiency in the project or in other words the skills of the testers in the project. Therefore, β may be considered as the learning index. The lower the value of βthe higher are the test efficiency and so better are the skills of the testing team. The scale parameter θ indicates the reciprocal of failure occurrence rate per fault or in other words reciprocal of rate of error detection per fault. Therefore, θ may be considered as the test intensity index. The lower the value of θ, the higher will be the test intensity. The location parameter γ indicates failure free time.
It is to be noted that the cumulative number of failures μ(t)= 0 for 0 < t ≤ γ
= a/c for t $\rightarrow \infty$

In the case of perfect debugging both the number of faults and failures detected at infinite time will be equal. If imperfect debugging is witnessed in the project, number of failures at infinite testing time will be greater. If efficient debugging is witnessed in the project, number of failures at infinite testing time will be lesser.

The failure intensity function, λ(t) is the rate of change of mean value function over time or the number of failures per unit time and hence it is the derivative of mean value function for failures with respect to time in Eq.(7). It is given by

$$\lambda(t) = a\left(\frac{\beta}{\theta}\right)\left(\frac{(t-\gamma)}{\theta}\right)^{\beta-1}\left[\exp\left(-c\left(\frac{(t-\gamma)}{\theta}\right)^{\beta}\right)\right] \qquad (8)$$

All the four models seem to provide satisfactory goodness of fit and predictive validity measures. However the models have 3, 4, 4 and 5 parameters respectively, Hence usage of these models would require software tool and for parameter estimation.

III. SRGM Tools

Commercial tools are available Off-The-Shelf (COTS) which can be used for parameter estimation of Software Reliability Growth Models. The characteristics of some of the COTS tools are briefly discussed below.

1. **SPSS Statistics** is a software package used for statistical analysis, the software was originally named the **Statistical Package for the Social Sciences** (**SPSS**). Later the expansion of the acronym was changed to **Statistical Product and Service Solutions** to reflect the growing diversity of the user base SPSS is a widely used program for statistical analysis in social science. It is also used by market researchers, health researchers, survey companies, government, education researchers, marketing organizations, data miners, and others.

2. **NCSS** is a computer program for statistical/data analysis created in 1981. NCSS LLC is the name of the company that produces NCSS. NCSS LLC specializes in providing statistical analysis software to researchers, businesses, and academic institutions.

3. **MATLAB** (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. Developed by Math Works, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

While the general purpose tools could be used for modelling, a dedicated tool will help the user to understand and use it efficiently. Hence a tool was developed using open ware Systems.

The Tool is developed with Open source software like JavaFX and R. JavaFX is a software platform for creating and delivering rich internet applications (RIAs) that can run across a wide variety of devices.

R is a free software programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls and surveys of data miners are showing R's popularity has increased substantially in recent years. R is a GNU project. The source code for the R software environment is written primarily in C, Fortran, and R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. R uses a command line interface; however, several graphical user interfaces are available for use with R.

JavaFX is used for Graphical User Interface (GUI), which acts as the front end for the users to manipulate the data, R is used as backend which performs as the Compiler Engine where the java sends given data to the REngine which can compile and execute rscripts and retrieves the results obtained from the REngine and yet to be displayed.
This combination was chosen meticulously by considering the following factors,

- Shorter memory requirements
- Higher Speed of Execution
- Platform Independence

IV. Analysis and design of tool

A. *Software*

R Platform

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed.

B. *Functional Requirements*

- **Rserve:** Rserve itself is the server that is a program that responds to requests from clients. It listens for any incoming connections and processes incoming requests. It is required to have R-1.5.0 or higher Version installed on user system in order to be able to use Rserve.

Framework:

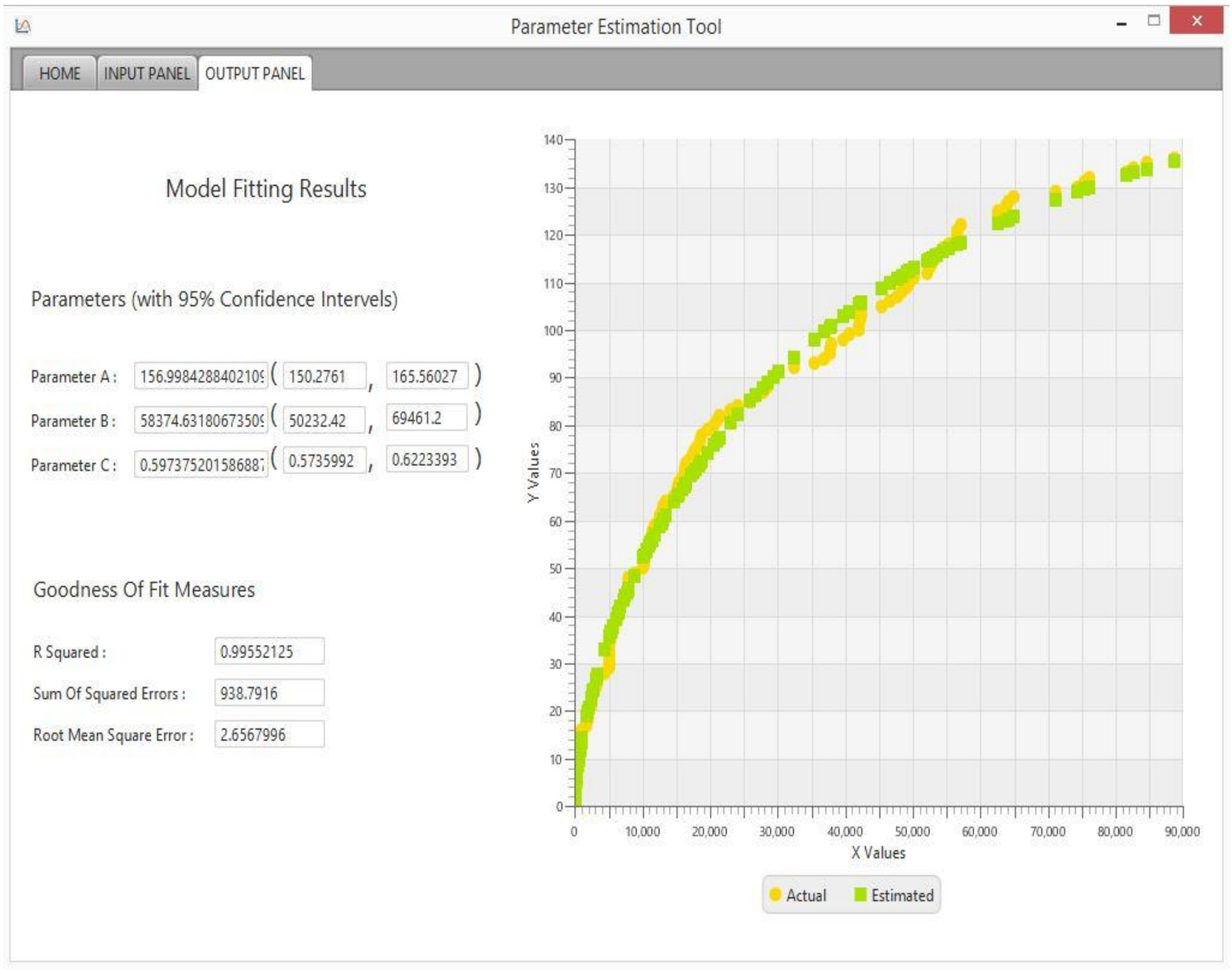
- **REngine:** Rengine class is the interface between an instance of R and the Java Virtual Machine (VM). Due to the fact that R has no threading support, you can run only one instance of R with in a multi-threaded application. There are two ways to use R from Java: individual call and full event loop.
- **REngine Java Client:** Java application (JDK 1.4 or higher) to access an Rserve. The suite is written entirely in Java. It provides automatic type translation for most objects such as int, double, arrays, String or Vector and classes for special R objects such as RBool, RList etc

  Pros: easy to setup, you don't need to initialize R or link against any R library, can run in a different machine, Enhanced interaction with R. Easier code Generation.
  Cons: based on TCP/IP (a server is running), no call-backs from R.

C. *Overall Design*

Here, in this system the structure of the system is divided into two parts: Input Panel and Output Panel.

- **Input Window:** A screen shot of Input Window is given in Figure 1. The user should enter the dataset in X and Y values and give Start, Minimum and Maximum values in Expected Range of Values, and selection of model is done by navigating through tabs inside the Input panel.


Parameter Estimation Tool

HOME | INPUT PANEL | OUTPUT PANEL

Model Fitting Results

Parameters (with 95% Confidence Intervels)

Parameter A :	156.998428840210	(150.2761 ,	165.56027)
Parameter B :	58374.6318067350	(50232.42 ,	69461.2)
Parameter C :	0.597375201586887	(0.5735992 ,	0.6223393)

Goodness Of Fit Measures

R Squared :	0.99552125
Sum Of Squared Errors :	938.7916
Root Mean Square Error :	2.6567996

Figure 1: Screenshot of Input Panel

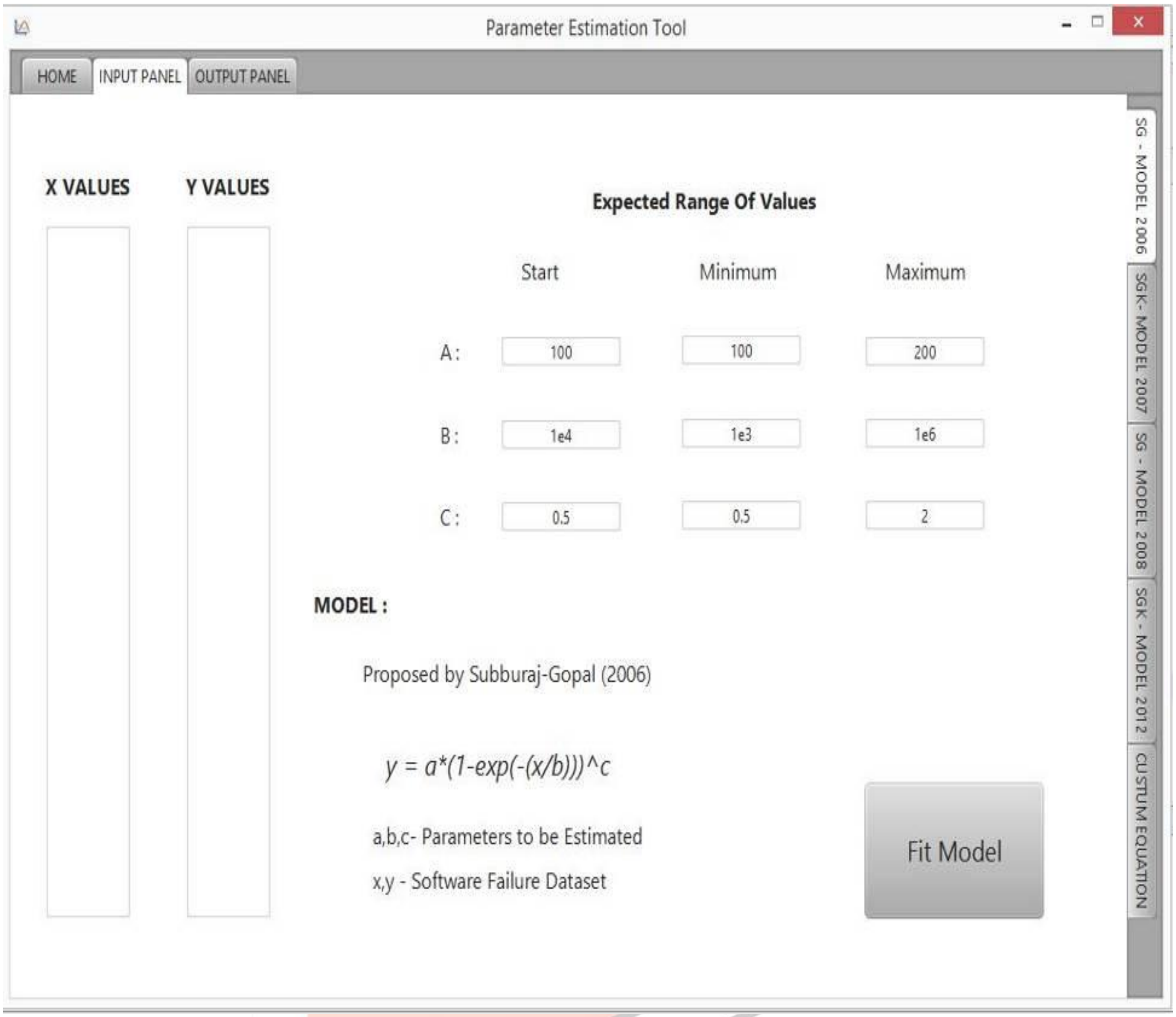
Models predefined in SRGM Tool:

- (Subburaj – Gopal) SG Model 2006
- (Subburaj – Gopal and Kapur) SGK Model 2007
- (Subburaj – Gopal) SG Model 2008
- (Subburaj – Gopal and Kapur) SGK Model 2012

Above mentioned four models are explained in detail at Flexible SRGMs (section II).

Musa [11], Software Reliability Dataset is used for testing and debugging the tool.

- **Output Window:** After Pressing Fit Model. It switches the tab to Output window, Earlier it was disabled, Parameters, Goodness of Fit (GoF), and plot for the actual and estimated values are plotted as shown in Figure 2.



Expected Range Of Values

	Start	Minimum	Maximum
A :	100	100	200
B :	1e4	1e3	1e6
C :	0.5	0.5	2

MODEL :

Proposed by Subburaj-Gopal (2006)

y = a*(1-exp(-(x/b)))^c

a,b,c- Parameters to be Estimated

x,y - Software Failure Dataset

Figure 2: Screenshot of Output Panel

SRGM Tool Features:

SRGM tool is an integrated suite of software facilities for software failure data manipulation, calculation and graphical display and Model parameter Estimation It includes:

- An effective data handling and storage facility
- Graphical facilities for data analysis and display on-screen for monitoring
- Higher Speed of Execution
- Platform Independence

V. Conclusion

While a number of commercial software tools are available in the market, need for a dedicated and expandable open source tool is felt by the Software Reliable Professional. In this paper the necessity and features of the SRGM tool developed by the authors are briefly discussed. We are confident that reliability engineers will find it to be appropriate for their use and learn to use them in a smarter and efficient manner.

References

[1] Xie, M. Software Reliability modelling. World Scientific, Singapore. 1991.

[2] S. Brocklehurst, et al., Recalibrating software reliability models, IEEE Trans. Software Eng. 16 (4), pp. 458–469, 1990.

[3] Musa J.D., and Ackerman A.F. Quantifying software validation: When to Stop Testing? IEE Software, pp19-27, 1989.

[4] Goel, A. L. and K. Okumoto. Time Dependent Error Detection Rate Model for Software Reliability and Other Performance Measures. IEEE Transactions on Reliability, R28 (3): 206-211, 1979.

[5] Yamada, S. Et al S-Shaped Reliability Growth Modeling For Software Error Detection, IEEE Transaction Reliability, Vol R-32, No.5, pp475-478, 1983.

[6] Goel, A.L, Software Reliability Models: Assumptions, Limitations and Applicability. IEEE Transactions of Software Engineering, vol. SE 11(12), 1411 -1423 1985.

[7] Subburaj R. and Gopal G. l, Generalized Exponential Poisson Model for software reliability growth, International Journal of Performability Engineering, Vol.2, NO.3,pp.291-301,July,2006

[8] Subburaj, R. and G. Gopal. Software Reliability Growth Model Addressing Learning. Journal of Applied Statistics, 35(10):1151 – 1168, 2008.

[9] Subburaj, R., G. Gopal and P.K. Kapur. A Software Reliability Growth Model for Vital Quality Metrics. South African Journal of Industrial Engineering, 18 (2): 93-108, 2007.

[10] Subburaj R, Gopal G and Kapur P.K, A Software Reliability Growth Model for Estimating Debugging and Learning Indices, International Journal of Performability Engineering Vol.8,NO.5, pp. 539-549, September 2012

[11] Musa J. D., DACS Software Reliability Dataset, Data & Analysis Centre for Software. 1980, www.dacs.dtic.mil/databases/sled/swrel.shtml, downloaded on Sept. 19, 2005.