www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 11 Nov. 2016, Page No. 18886-18891

Software Reliability Testing Tools: An Overview and Comparison

Manohar Singh
Research Scholar, Department of Computer Science
OPJS University, Churu, Rajasthan
ms_hodcs@yahoo.com

Abstract: *Software reliability has been regarded as major quality attribute and still there are very few available standardized tools. Software reliability is such a significant factor in finalizing the overall quality of software, thus it must be estimated. In this paper we will discuss various software reliability metrics. A well designed metrics can helps in detection and correction of requirement faults that will guide in avoidance of error in later stage of software development. A software reliability growth model is one of the basic techniques used to evaluate the software reliability quantitatively. The software growth model is required to have a significant performance in term of goodness of fit, reliability etc. In this paper we will summarize some existing software reliability tools such as CASRE, SMERFS, SOFTREL, SOREL etc. Overall the paper will provide various ways to enhance software reliability.*

1. Software Reliability:

It is believed that software is always correct and once it runs correctly it will be work rightly forever this gave rise to the idea of Software Reliability. According to ANSI, ―Software Reliability is defined as the probability of software operation that is failure-free for a particular period of time in a particular environment. The probability of failure-free software operation for a specified period of time in a specified environment is known as Software reliability. In highly complex modern software systems, reliability is most significant feature. Software Reliability reduces software failures during the development of software and software quality control in the complex modern software systems. Software reliability can also be defined for software as the probability of carrying out without failures for some specific interval of time. A fault is a defect that causes or can potentially cause the failure of software when it is executed [3]. Unpredictability of any software comes due to the failures or occurrence of faults in the system. As software does not "exhaust" or "deteriorate", as a mechanical or an electronic system does, the changeability of software is mainly due to bugs or design faults in it. Reliability is a probabilistic measure that believes that the incidence of failure of software is a random phenomenon.

Therefore we can say that reliability is a important aspect in deciding the general excellence of software. Researchers shift towards the estimation of Software Reliability which gives rise to the variety of software reliability estimation tools. There are many software reliability measuring tools available for users to implement one or more of the software reliability model to a software development purpose and to launch the applicability of a particular model to a set of failure data. A key issue in modeling software reliability lies in the accessibility of present available tools. Almost all the tools have command-line interfaces, and use least of the high-resolution displays that would permit the creation of menu-driven or direct-manipulation user interfaces [5]. In measuring software reliability, it is helpful to see high-resolution displays of these quantities, as well as growing number of errors and the results of statistical methods used to establish whether the model being used is feasible for the present project[7][4].

2. Reliability Process

The reliability method in large terms is a model of the reliability-oriented features of software development, operations and maintenance. The set of life cycle actions and artifacts, together with their attributes and interrelationships that are combined together to reliability, consists of the reliability process. The main artifacts of software life cycle are documents (Software Requirement Specification), project manuals, reports, design or plans, code, configuration data and test data. Software reliability is dynamic and continuous process. In a new product, it begins at a low figure with respect to its new proposed usage and eventually reaches a stage near unity at its development. The correct value of product reliability however is never accurately known at any point in its life span. These tasks are performed by the Software Reliability Engineering (SRE) tools (see Figure 1.):

- Gathering failure information and test time detail
- Calculating estimates of model parameters using the information obtainable.
- Testing to fit a model with the collected information.

DOI: 10.18535/ijecs/v5i11.29

- Selection of a model to make prediction of left over faults etc.
- Applying the selected model as per set criteria.

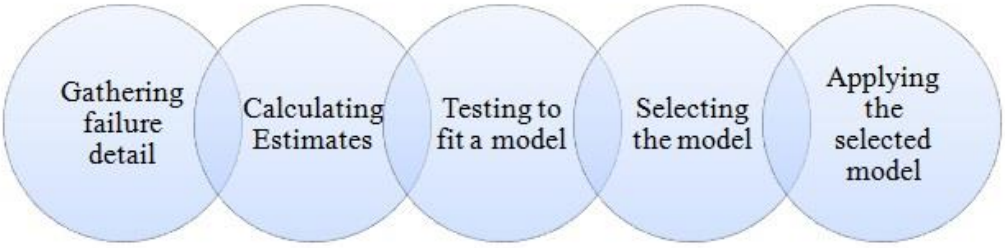

Figure 1. Tasks in SRE Tools

3. Software Reliability Testing

To enhance the performance of software and software development process, a comprehensive evaluation of reliability is necessary. Testing software reliability is significant because it is of immense use for software developers. Reliability testing is of many types. Some of the major types of software reliability testing are:

Feature test: Feature testing evaluates the features provided by the software. Feature testing is conducted in the following steps:

- Each operation in the software is executed once.
- Operation and their interactions are reduced.
- Each and every operation is checked for its proper implementation.

After the feature test, load test is conducted.

Load test: Load test is conducted to evaluate the performance of the software under utmost work load. Software's generally performs better up to certain extent of workload, after that there are significant degradation in software response time. For example, a web site can be tested to see how many concurrent users it can sustain without its performance degradation. This type of testing mainly helps in checking the reliability of Databases and Application servers. Load testing also requires software performance testing, which evaluates how fit some software performs under workload.

Regression test: Regression testing is used to check if any new bugs have been developed through fixing of previous bugs. Regression testing is often conducted after change or modification in the software features. This testing is cyclic, depending upon the length and features of the developed software.

4. Software Reliability Metrics

Software Reliability Measurement is not a precise science. Though annoying, the hunt of quantifying the software reliability has never ceased. Until now, we have no good quality techniques for measuring software reliability. Measurement of software reliability remains a tricky problem because we don't have a complete idea of the nature of Software. There is no apparent description to what aspects are linked to software reliability. Reliability metrics are used to convey the reliability of the software product. The selection of which metric is to be used depends upon the type of system to which it applies & the requirements of the application field. We cannot find a suitable way to measure software reliability, and most of the aspects related to software reliability. It is attractive to measure somewhat related to reliability to reveal the features, if we cannot measure reliability straightforwardly. Some reliability metrics which can be used to calculate the reliability of the software product are:

- **MEAN TIME TO FAILURE (MTTF):** MTTF is defined as the time interval between the consecutive failures. An MTTF of 200 means that one failure can be expected every 200 time units. The time units are totally dependent on the system and it can still be specified in the number of transactions. MTTF is applicable for systems with extensive transactions. For example, it is appropriate for computer aided design systems where a designer will work on a plan for several hours as well as for Word-processor systems.
- **MEAN TIME BETWEEN FAILURES (MTBF):** We can combine Mean Time to Failure and Mean Time to Repair metrics to get the Mean Time between Failures metric.

$$MTBF = MTTF + MTTR$$

  Thus, a MTBF of 200 indicates that once the failure occurs, the next failure will probable occur only after 200 hours. In this case the time measurements are real and not the execution time as in the case of MTTF.
- **RATE OF OCCURRENCE OF FAILURE (ROCOF):** It estimates the number of failures occurring in unit time interval. The no. of unexpected events over a particular time of operation. ROCOF is the frequency of occurrence with which unpredicted

behavior is likely to occur. An ROCOF of 0.04 means that four failures are likely to occur in each 100 operational time unit steps. It is also known as failure intensity metric.

- **MEAN TIME TO REPAIR (MTTR):** Once the failure occurs in the software, sometime it is required to fix the bug. MTTR measures the average time it is required to track the errors which is causing the failure and time required to fix the bug.
- **PROBABILITY OF FAILURE ON DEMAND (POFOD):** POFOD is defined as the possibility that the software system will fail when a service is requested to it. POFOD is the number of system breakdown given a number of systems inputs. POFOD is the likelihood that the system will fail when a service request is made. A POFOD of 0.2 means that two out of a ten service requests may result in failure. POFOD is a significant measure for safety critical systems. It is appropriate for protection systems where services are demanded occasionally.
- **AVAILABILITY (AVAIL):** Availability (AVAIL) is the probability that the system is accessible or available for usage at a specific time. It accounts the repair time and the restart time for the software system. An availability of 0.990 means that in every 1000 time units, the system is expected to be available for 990 of these. The percentage time that a system is available for use, considering into account deliberate and unintended downtime. If a system is down an average of five hours out of 100 hours of operation, its AVAIL is 95%. [1][8]

5. Software Reliability Growth Model

A Software Reliability Growth Model is one of the elementary techniques to calculate software reliability quantitatively. All the reliability growth models are based upon the hypothesis that the reliability of a program is a function of the number of faults that it contains. The Software Reliability Growth Model required a good performance in terms of goodness-of-fit, certainty. In order to estimate as well as to predict the reliability of software systems, failure data need to be properly measured by various metrics during software development and operational phases. Such models use statistical techniques to observe the failures during software testing and its operations to predict the product's reliability. The data used in the growth model is taken from where the software will be deployed. Any software required to function reliably must undertake widespread testing and debugging. This process can be a costly and time consuming, and developers require exact information about how software reliability grows as a result of this process in order to successfully manage their budgets and projects. Such type of techniques permit managers to precisely allocate time, cost, and other resources to a project, and assess when software has reached a position where it can be released with some assurance in its reliability. A number of analytical models have been derived to check the problem of software reliability measurement.[5][2] These approaches are indicated below:-

- **Times between Failures Models (TBF):** In this class of models, the process under study is the time between failures. Estimates of parameters are derived from the detected values of times between the failures and estimates of software reliability, mean time to next failure are then obtained from fitted model.
- **Failure Count Models (FC):** The use of this class of models is in the number of faults or failures in particular time intervals rather than in times between two failures. The failure counts are implied to follow a known stochastic process with a time dependent distinct or constant failure rate. Parameters of the failure rate can be estimated from the observed values of failure counts or from failure times.
- **Fault Seeding Models (FS):** The basic approach in this type of model is to recognize a known number of faults in a program which is understood to have an unidentified number of indigenous faults. In this case program is tested and the practical numbers of seeded and local faults are counted. From these, an estimate of the fault content of the program proceeding to seeding is obtained and used to review software reliability and various other relevant measures.
- **Input Domain Based Models (IDB):** This model is used to generate a set of test cases from an input distribution which is understood to be agent of the operational practice of the program. This type of distribution is difficult to obtain. Because of the complexity in obtaining this distribution, the input domain is divided into a set of correspondence classes, all of these are usually associated with a program pathway [9][6].

Times Between Failures Model (TBF)	• Independent times between failures. • Equal probability of the exposure for each fault. • Faults are corrected after each occasion • New faults introduced during rectification.

Fault Count Model (FC)	• Testing intervals are free of each other. • Testing during intervals is logically homogenous. • Numbers of bugs detected during non-overlap intervals are independent of each other.
Fault Seeding Model (FS)	• Seeded faults are randomly scattered in the program. • Indigenous and seeded faults have equal chance to be detected.
Input Domain Based Model (IDB)	• Input profile distribution is known. • Random testing is performed. • Input domain can be divided into equivalent classes.

Table 1: List of Key Assumptions by Model Category

6. Software Reliability Testing and Growth Testing Tools

(I) CASRE (Computer-aided software reliability estimation tool)

CASRE stands for Computer Aided Software Reliability Estimation. CASRE tool is used as a software reliability modeling tool that addresses the ease-of-use issue and other issues. This CASE tool is built on based upon existing software reliability models, called component models. The main feature of this tool over others such kinds of tools is its varied reliability estimations under a prototype which linearly combines the component models. CARSE features an enhanced graphical user-interface which significantly facilitates the tiresome application process for software reliability estimation. CASRE is actually an expansion of the public-domain tool SMERFS, and is planned to use both in a DOS- Windows environment and as well as in UNIX - windows environment. In this tool users are guided through the selection of a group of breakdown data and running a model. Modeling results are shown in a graphical as well as in tabular form. Apart from this, CASRE contains various other functionalities. This tool is accomplished to increase forecast accuracy by mixing the results of various other models in a linear way. Moreover, CASRE also allows managers to define their own combinations and documentation them as component of the CARSE tool configuration.

(II) SMERFS (Statistical modeling and estimation of reliability functions for software)

Another software reliability prediction tool is Statistical Modeling and Estimation of reliability functions for Software (SMERFS). This is a widely used and a accepted software application for evaluation of test data for checking failure rate and fault detection rate forecast. The input to SMERFS is a collection of values comprises either of the time between detection of defects or the number of defects detected per time period. SMERFS then uses maximum probability method or least squares methods to approximation the parameters used for one or more of these models. SMERFS output comprises the parameter estimates, a measure of the goodness-of-fit and predicted values using the chi-squared distribution.

(III) SOFTREL

The software reliability process simulator SOFTREL contains the effects of interrelationships among activities, and combines all events as piecewise-Poisson Markov processes with clearly defined event rate functions. The documentation simulated by Software Reliability process Simulator consists of requirements, design, interface specifications, and other entities whose absence or defective nature can cause errors into subsequently produced code. Integration and test procedures, management plans, and other documentation, when deemed not to correlate directly with fault generation, are excluded. The assumption is that the likelihood of a fault at any particular time increases proportionately to the amount of documentation missing or in error. Requirements analysis and design activities are currently united in the document construction and integration phases in SOFTREL. All errors occur either in proportion to the amount of latest and reused documentation, to the amount that was changed, deleted, and added, or to the number of errors that were reworked.

(IV) SRMP (Statistical modeling and reliability program)

The SRMP was developed by the Reliability and Statistical Consultants Ltd. of UK in 1988. It is a command-line-oriented tool developed for an IBM PC/AT and also UNIX based workstations. SRMP contains nine models. It uses the utmost likelihood estimation technique to estimate the model parameters, and includes the following reliability indicators:

- Reliability function
- Failure rate
- Mean time to failure
- Median time to failure.
- The model parameters for each model.

SRMP requires an ASCII data file as an input. The file contains the name of the project, the number of failures involved in the reliability analysis, and the inter failure times of all the failures. The input file also describes the initial sample size used by SRMP for the initial fitting of each reliability model to the data. The failures which remained are used by SRMP for accessing a reliability model's prediction accuracy.

Parameters → TOOLS ↓	Language	Reliability Rate	Total Failure	Remaining Failure	User Assistance	Graphics	No. of models supported
CASRE	FORTRAN	Y	Y	Y	Y	Y	16
SMERFS	FORTRAN	Y	Y	Y	Y	Y	12
SOFTREL	C	Y	Y	Y	Y	Y	2
SRMP	FORTRAN	Y	Y	Y	Y	Y	9
SOREL	PASCAL	Y	Y	Y	Y	Y	4

Table 2: Comparison of Various Reliability Tools

(V) SoREL (Software reliability analysis and prediction)

SoRel is widely used tool for Software Reliability analysis and prediction. It comprises two parts. They allow reliability growth tests and applicability of reliability growth models. SoREL allows two types of failure data processing i.e. inter-failure data and failure intensity data. The main actions that can be evaluated are: the mean time to next failure, the intensity function, the collective number of failures and the residual failure rate of the software.

In SoREL Four type of reliability growth tests are offered: arithmetical mean, Laplace test, Kendall test and Spearmann test (for both failure intensity data and inter-failure). Four reliability growth models are applied to allow different kinds of behavior to be modeled:

- a decreasing failure rate which resulting to zero when time tends to infinity
- a decreasing failure rate which resulting to a non zero value
- an increasing failure rate which is followed by a decreasing failure rate
- a model can be analyzed according to its reproductive capability

The results in SoREL are available into two forms:

- Immediately on the monitor (numerical results and curves),
- in the form of files, which can be used by other applications (Excel, Word…), numerical results and curves [3].

7. Conclusion

Software reliability is a vital research area and software reliability is a very significant part of software quality. Software reliability metrics can be used to assess present software reliability and forecast future. Achieving software reliability is a key issue in a software industry. But this reliability is hard to achieve due to software complexity. Software reliability can be enhanced but complete reliability is still distinct. This paper discussed various software reliability metrics and its uses. Apart from this, the paper discussed several software reliability testing tools and software reliability growth testing tools available in the market. Use of these tools can significantly increase the reliability of software drastically.

Reference

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

[2] Gaurav Aggarwal, Dr. V.K Gupta, "Software Reliability Growth Model", International Journal of Advanced Research in Computer Science and Software Engineering, Jan 2014.

[3] Dileep Sadhankar, Dr. Ashish Sasankar, "An overview and comparison of Software Reliability tools", IOSR Journal of Computer Engineering (IOSR-JCE)

[4] Vaibhav E. Pawar, Amol K. Kadam , "Analysis of Software Reliability using Testing Time and Testing Coverage", International Journal of Advance Research in Computer Science and Management Studies, May 2015

[5] Razeef Mohd., Mohsin Nazir, "Software Reliability Growth Models: Overview and Applications, Journal of

Emerging Trends in Computing and Information Sciences, Sep 2012

[6] C. Stringfellow, A. Amschler Andrews, "An Empirical Method for Selecting Software Reliability Growth Models"

[7] Mehraj – Ud - Din Dar, S. M. K. Quadri, "Improving Software Reliability using Software Engineering Approach-A Review" Aasia Quyoum, International Journal of Computer Applications, Volume 10– No.5, November 2010

[8] Gurpreet Kaur, Kailash Bahl, "Software Reliability, Metrics, Reliability Improvement Using Agile Process", IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 3, May 2014.

[9] D.Swamydoss, Dr. Kadhar Nawaz, "Enhanced Version of Growth Model in Web Based Software Reliability Engineering", JGRCS 2010