METRICS AND MEASUREMENT ISSUE OF SOFTWARE RELIABILITY

Mukesh Bansal*

Adesh Kr. Pandey*

Sangeeta Arora*

ABSTRACT

Software reliability is the probability of failure free software operation for a specified period of time in a specified environment. Software reliability, however, is generally accepted as the key factor in software quality since it quantifies software failures. Reliability is a by-product of quality , and software quality can be measured. Current methods to measure the reliability of software are usually focused on large server based products. For such products unique issue arise in obtaining the failure and population data and in analysing this data to determine reliability. In this paper we review some of the key issue in measuring reliability of such software products and software metrics to improve the reliability and quality of the software products.

*Research Scholar, Department of Computer Applications, Makhanlal Chaturvedi National University of Journalism and Communication, Bhopal.

1. INTRODUCTION

Software reliability engineering is centred around a very important software attribute: reliability. Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is an important attribute of software quality, together with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation. Software failures may be due to errors, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software or other unforeseen problems. Software Reliability is hard to achieve, because the complexity of software tends to be high. While the complexity of software is inversely related to software reliability, it is directly related to other important factors in software quality, especially functionality, capability, etc. The primary goal of software reliability models is to assess current reliability and forecast future reliability, based on rational assumptions for the application of statistical inference techniques to the observed failure data. Many software reliability growth models have been proposed, which estimate the reliability of a software system as it undergoes changes through the removal of failure causing faults. For a general purpose software product, once the product is released to a large number of users, the possibility of providing an accurate measure of reliability, shortly after product release, becomes more feasible as large amount of failure data can be captured from the end users through various data collection processes. In this paper we first discuss reliability metrics to improve the software reliability and then to discuss some key issue to measure the reliability of software products. [3][7]

1.1 Definition

Software Metrics: Software Metrics can be defined as "The continuous application of measurement based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products". The most established area of software metrics is cost and size estimation techniques.

Software metrics are measures of the attributes of software products and processes. They are increasingly playing a central role in the planning and control of software development projects. Either the requirements specification or the software product design can be used to estimate the size and complexity of the planned software product. These measures can then be used to estimate the time needed for development and testing before product release.

Software Reliability: As per IEEE, standard "Software reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a special period of time".[4]

Software measurement: It is a quantified attribute of a characteristic of a software product or the software process

2. SOFTWARE RELIABILITY METRICS

Software reliability metrics are used for software reliability evaluation and assurance. In this section we discuss metrics that makes software more reliable.

2.1 Requirement Reliability Metrics:

Requirement specifies the functionality that must be included in the final software. It is very important that the requirement should be collected such that there is no misunderstanding between the developer and the client. For high software reliability the requirement must be structured, complete and easy to apply. Requirement specification should not include phrases such that to be determined or to be added because it reflects the negative impact on the design of the system. The requirement must contain the valid structure to avoid the loss of valuable information. Requirement reliability metrics evaluates the above said factors of the requirement document.

2.2 Design and Code Reliability Metrics

Software Reliability is hard to achieve, because the complexity of software tends to be high. The complexity has a direct impact on overall quality. Software size is thought to be reflective of complexity, development effort and reliability. Lines of Code (LOC), or LOC in thousands (KLOC), is an intuitive initial approach to measuring software size. But there is not a standard way of counting. Function point metric is a method of measuring the functionality of a proposed software development based upon a count of inputs, outputs, inquires, and interfaces. Complexity-oriented metrics is methods of determining the complexity of a program's control structure, by simplify the code into a graphical representation.

2.3 Testing Reliability Metrics

Test coverage metrics are a way of estimating fault and reliability by performing tests on software products, based on the assumption that software reliability is a function of the portion of software that has been successfully verified or tested. Fault and failure metrics is able to determine when the software is approaching failure-free execution. Test strategy is highly relative to the effectiveness of fault metrics, because if the testing scenario does not

cover the full functionality of the software, the software may pass all tests and yet be prone to failure once delivered. Usually, failure metrics are based upon customer information regarding failures found after release of the software.

3. MEASURING RELIABILITY

Measurement is the key to achieve high reliability software. To make software more reliable the focus should be to identify the knowledge and skills that are required to advance the measurement component of software engineering rather than focus on coding phase of the development process In this section first we identify knowledge requirement in software reliability measurement and later on we discuss some of the key issues faced when measuring the reliability of a software product.

3.1 **Identifying Knowledge Requirement Issue**

1. Goals: what reliability goals are specified for the system? We analyze the software quality goals of the organization .This analysis could consist of interviewing key personnel in the organization and examine documentation that address reliability goals. Metrics are identified and data collection plans are developed for satisfying reliability goals.
2. Cost: what is the cost of achieving reliability goals? As the complexity of the software increases, it will increase the level of testing which reflects the cost of the software.
3. Context: what application and organization structure is the system and software to support? This function involves analysing, identifying and classifying the application domain.
4. Operation Profile: An operation profile is a quantitative characterization of how a system will be used. What are the critically and frequency of use of the software component. It identifies operations, their criticality and the relative frequency of occurrence.
5. Model: what is the feasibility of creating and using an existing reliability model for assessment and prediction and how can the model be validated. software reliability modelling is a set of techniques that apply probability theory and statistical analysis to predict the reliability of software products, both quantitatively and objectively. The primary goal of software reliability models is to assess current reliability and forecast future reliability, based on rational assumptions for the application of statistical inference techniques to the observed failure data.

6. Data Requirement: what data are needed to support reliability goal. What data is to collect should be determined, assumptions that will be made about the data are stated, and the flow of data from point off collection to evolution of metrics, tools and their usage are described.[1]

   These knowledge requirements issue are summarized in table-1

3.2. Measurement Key Issue

Failure Classification

As reliability is concerned with the frequency of different types of failures we need to have a clear and unambiguous classification of failures. For a modern software product, we suggest that failures be partitioned at the top level as *unplanned events*, *planned events*, and *configuration failures*. Unplanned events are traditional failures like crash, hang, incorrect or no output, which are caused by software bugs. Planned events are those where the software is shutdown in a planned manner to perform some housekeeping tasks. Configuration failures occur due to problems in configuration setting.

The Population Size

A key data we need for determining reliability is the population size, that is, how many units of the product are in operation. We propose that for determining reliability a (random) sample, called the *observed group*, of the population size be taken. With this identified observed group, failure data will be recorded only for this group of users. Regular statistical techniques can be used to determine the sample size such that the final result is accurate to the degree desired. It has been observed that in many cases failure rate of units decrease in the initial stages as users stabilize their configuration and learn to avoid failure causing situations. By fixing the sample relatively early, we avoid the problem of mixing failure rates of old and new units, and can easily determine the steady state reliability.

Obtaining Failure Data

For reliability computation, we need a mechanism to collect failure data, where the failures are occurring on system used by users distributed around the world. The users in the observed group are periodically asked to fill a form to report failures they have experienced in the last 24 hours. This form can be a simple, with check boxes for each failure type, and its submission can be easily automated. The most accurate data collection for the observed group will occur if the data is collected and reported automatically through proper instrumentation and triggers in the product.

Usage Time

For an accurate computation of reliability, the actual usage time of the product by the user needs to be determined to be able to calculate the failure rates. As a convenience, it is often implicitly assumed that the product is used, on an average, for the same amount of time every day by every user. With this assumption, the day count can be used for determining reliability.[2]

Software Reliability Measurement Process

Software reliability measurement is the application of statistical inference procedures to failure data taken from software testing and operation to determine software reliability. Software Reliability measurement process can be understands through flow chart given in figure 1.In this flow chart first, customer usage is quantified by developing an operational profile. Second, quality is defined quantitatively from the customer's viewpoint by defining failures and failure severity, by determining a reliability objective, and by specifying balance among key quality objectives e.g., reliability, delivery date, cost, etc. to maximize customer satisfaction. We then advocate the employment of operational profile and quality objectives to manage resources and to guide design, implementation, and testing of software. Moreover, we track reliability during testing to determine product release, using appropriate software reliability measurement tools. This activity may be repeated until a certain reliability level has been achieved.[5]

It can be seen from figure 1 that there are four major components in this software reliability measurement process, namely,

(1) Reliability objective,

(2) Operational profile,

(3) Reliability modeling and measurement,

(4) Reliability validation.

A reliability objective is the specification of the reliability goal of a product from the viewpoint of the customer. If a reliability objective has been specified by the customer, that reliability objective should be used.

The reliability of a software based products depends on how the computer and other external elements will use it. [6] The Operational profile is a set of disjoint alternatives of system operation and their associated probabilities of occurrence. The construction of an operational profile encourages testers to select test cases according to the system's operational usage, which contributes to more accurate estimation of software reliability in the field.

Reliability modeling is an essential element of the reliability estimation process. It determines if a product meets its reliability objective and is ready for release. Despite the existence of more than 40 models, the problem of model selection and application is manageable. Experience has shown that it is sufficient to consider only a dozen models, including Jelinski-Moranda Model, Generalized Poisson Model, Goel-Okumoto Model, Musa Basic Model, Musa-Okumoto Model, Schneidewind Model, Non-Homogeneous Poisson Process Model, Delayed SShape Model, and Littlewood-Verrall Bayesian Model, etc. Using these statistical methods, "best" estimates of- reliability are obtained during testing. These estimates are then used to project the reliability during field operation in order to determine if the reliability objective has been met. This procedure is an iterative process since more testing will be needed if the objective is not met. When the operational profile is not fully developed, application of a test compression factor can assist in estimating field reliability.

Finally, the projected field reliability has to be validated by comparing it with the observed field reliability. This validation not only establishes benchmarks and confidence levels of the reliability estimates, but also provides feedback to the software reliability measurement process for process improvement and better parameter tuning.

4. CONCLUSION

Software reliability is the key part of software quality. software reliability models can be used to assess current reliability and forecast future reliability. At each phase of the development life cycle , metrics can identify the areas where the problems or error occurs. so by using the software reliability metrics we can increase the reliability of the software. In this paper we review some of the key issue that arises when measuring reliability of mass market product and also review some issue that are based on knowledgeable requirement to increase the reliability of the software product. It will therefore be best if systems are put in place for measurement to have the ability to provide information that can help reliability improvement.

5. REFERENCES

1. Norman F,Schneidewind,"Software ReliabilityMeasurement", Reliability review, The R & M Engineering Journal, Volume 23 Number 2, June 2003
2. P. Jalote, B. Murphy, M. R. Garzia and B. Errez. "Measuring Reliability of Software Products". ISSRE 2004 Conference. Saint- Malo, Bretagne, France, 2004.
3. Deepak Pengoria, Saurabh Kumar, " A Study on Software Reliability Engineering Present Paradigms and its Future Considerations "*School of Computing Sciences, VIT University, Vellore, India*

4. IEEE Standards 610.12-1990 Gloassary of Software Engineering Technology
5. Michael R Lyu," An Integrated Approach to Achieving High Software Reliability
6. Lalit Prasad, Ankur Gupta,Sarita Badoria,"Measurement of Software Reliability using Sequential Bayesian Technique" Proceedings of the World Congree on Engineering and Computer Science 2009 Vol-I WCECS 2009, San Frncisco, USA
7. Rosenberg, T Hammer, shaw,," software metrics and reliability", Proceedings of the 9th International…, 1998 - Citeseer

Table 1: Knowledge Requirements in Software Reliability Measurement

Issue	Function
1. **Goals:** What reliability goals are specified for the system?	**Analyze** reliability goals and **specify** reliability requirements.
2. **Cost:** What is the cost of achieving reliability goals?	**Evaluate** economics of reliability goals.
3. **Context**: What application and organizational structure is the system and software to support?	**Analyze** the application environment.
4. **Operational profile**: What are the criticality and frequency of use of the software components?	**Analyze** the software environment.
5. **Models**: What is the feasibility of creating or using an existing reliability model for assessment and prediction, and how can the model be validated?	**Model** reliability and **validate** the model.
6. **Data requirements**: What data are needed to support reliability goals?	**Define** data type, phase, time, and frequency of

Figure -1: **Flow Chart of software Reliability Measurement Process**