International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol. 3, Issue. 6, Nov - Dec. 2013 pp-3380-3384 ISSN: 2249-6645

Aggrandize the Reliability by Bug Retrieval (ARBR)

Ms. J. Arthy
Assistant Professor, Dept. Of CSE, Rajiv Gandhi college of Engineering and Technology, Puducherry, India

Abstract: *A complex software system has numerous defect prone tasks. There are many reasons for the defects in software. Many of these defects lead to failure of software. Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. No good quantitative methods have been developed to represent Software Reliability without tedious manipulation. Therefore, we need a method to enhance the reliability without complexity. We proposed an approach named as ARBR reduces the complexity and manipulation of software metrics. ARBR collects the possible bugs from various applications and maintains an audit for the same. When the new application detects the bug as an audit then the system gets restarted automatically where the executed applications and operations saved so as to recover things easily then the system again try the applications by eliminating the bugs. This method is evaluated in some projects where the bugs are eliminated and gives desirable performance. ARBR satisfies the software reliability requirements with balancing the risk of failures and cost.*

Keywords: *Software reliability, Audit maintenance, Bug analysis, Recovery, KLOC.*

I. INTRODUCTION

Nowadays, information processing systems and their software provide a good factor such as low cost, good control, compact processors, easy handling, etc.. This section discusses about the Software reliability, their parts and the various reasons.

1.1 SOFTWARE 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 different from hardware reliability because time is not a major constraint. It will not modify over time unless changed or upgraded happened frequently. There are many software quality features such as usability, maintainability etc.… Reliability is one of them and it is very hard to attain it because it leads to a high degree of complexity when the software application size is big and enormous. The complexity of an application is inversely related to reliability and directly related to quality. Good projects are emerging from good management such as time, cost and development.

Software reliability consists three parts: modeling, measurement and enhancement. Reliability modeling refers the optimized model which is ensured by a system testing. There are various estimation techniques to measure the reliability, enhancing the reliability is the process of increasing capability of software during testing and implementation.

There are various reasons behind software failures such as errors, interpretation faults, incompetence, testing and other problems. Design faults also affect the reliability of software. The measurement of software reliability completely depends on manipulation and calculation and so physical prediction is not possible. There are some worst situations where the error appears without any warning. For example, the inputs of a program also affect the software in the situations like redundancy, interference and overlapping. By analyzing the above issues the standard testing and large testing is examined to improve software reliability. But there are no standard methods other than some logic structures and calculations. There is not a simple method to measure software reliability. If the programmer or user doesn't understand the software system or application then it becomes very rigorous to measure it. Most of the software metrics not have a common definition or methods. There are many metrics taken into consideration such as product, process, fault and failure metrics. These metrics help the designer to measure reliability indirectly. There is no standard way of counting the application other than LOC (Lines Of Code) or LOC in thousands (KLOC). Software operating environments are different for every application and so it also affects the reliability .There are two issues which violate the software reliability. They are Control dependence and data independence. The hardware reliability is worn out but the software reliability is conceptual and document. To enhance software reliability first the measurement and improvement of the metrics are initialized. At the next step the cost, effort, time and other set of complexity metrics must be low. There are many real time examples of software failure such as accidental change of function when fault input to the system, encountered, misinterpretation of requirements etc., All the above discussed issues ensure that the reliability of a software is an unpredictable one. So there is a need for a good prediction method of software reliability. To enhance reliability the following steps to be done (i) standardizing data collecting methods (ii) documenting again and again (iii) inter-rater reliability. We propose a prediction reliability model in further sections.

II. RELATED WORK

Most reliability methods consider some assumptions and mathematical functions such as higher order exponential and logarithmic. There are numerous approaches can be used to increase the reliability. However, all of the methods are very tedious and rigorous to develop and implement by means of time and cost. In the literature, there are some works for attaining reliability models. Brown et al. introduced a cost model that determines the optimal number of software test cases based on a probabilistic model that incorporates the cost per test, the error cost, the number of software executions, and the estimated number of faults. The main disadvantage of this approach is that no distinction is made between different tests, and the fact that these different tests cover different possible faults. The main assumption used by these models in treating the fault correction process is that the rate of fault correction is proportional to the number of faults to be corrected, meaning that the expected cumulative number of corrected faults is proportional to the expected cumulative number of detected faults with constant delay. All the methods have some assumptions and calculations.

Eduardo Oliveria Costa et al. (2010) introduced a model by the method of Genetic Programming (GP) it gives the better reliability curve which is experimenting with time and coverage. Kapil Sharma et al. (2010) develop a deterministic quantitative model based on a distance based approach and then applied for evaluation, optimization and selection. Ahmed Patel et al. (2010) develop a technique for software application and reliability which is implemented with programming examples. Costa et al. (2010) follows a genetic programming approach for reliability modeling. J. Onishi et al. (2007) gives an improved surrogate method which solves redundancy and other problems which increases the reliability method. Yousif A. Bastaki (2012) developed an interactive method which gets the things from a user and increases the reliability modeling.

III. APPROACH

By analyzing the above models there is no single model which works in all environments efficiently. Some of them may be working well in a certain environment and not suited for other environments. To overcome the above drawbacks ARBR model is introduced where the manual calculation and complexity is very low and compatible with all environments. Many applications are examined by some tools and the errors and faults in the applications are experimented for ARBR-model. By analyzing and gathering these bugs it is possible to discover the errors in a new application. The main advantage of this work is to enhance reliability and redesign to another application. This method is generic considered until other methods. This method achieves high reliability through configuration and interactive and instantaneous development process. The below figure 1 give the overall description of ARBR model.

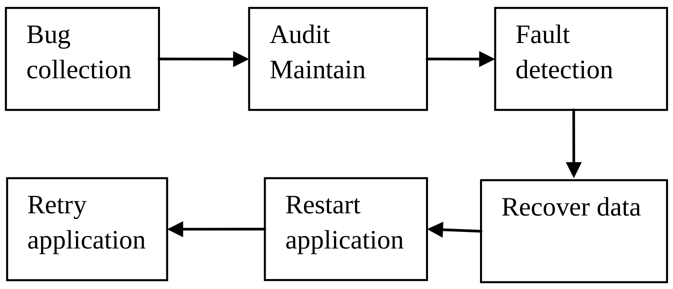

Fig 1 Overview of ARBR

The possible bugs are collected from various applications which help us to determine when the new software application faces the failure. The failures occurred in all types of system are gathered and maintain as a database. The collection is simple because the failures are collected during testing and some are reported by the users. The depth survey users are also given the needed details. Metrics determine the efficiency of the review collected from the user. For example, fault metrics used to measure reliability by means of failure density and Mean Time Between Failures (MTBF). Based on the above mentioned parameters there is a need to increase the reliability and quality. The database is maintained by any packages. So this method is compatible with all environments. When an application fails ARBR try to recover data and executed operations and then the system restarts automatically. The abrupt changes and disaster cannot affect this operation by the mode of recovery. During implementation when the application faces the collected defects in the database then the system is restarted automatically where the current operations and applications saved. In ARBR, It is very easy to retrieve the data from any defects. So the obtained and executed results are protected and we are able to retry the same application. This method has two phases (i) Prediction (ii) design. In the first phase the failure and other properties such as causes of failures are analyzed and in the second phase the application is accumulated. In the first phase some historical application is used and in the next phase predicts the reliability and estimate the same using any estimation techniques. The below diagram Fig 2 shows the method of collecting faults from various projects.

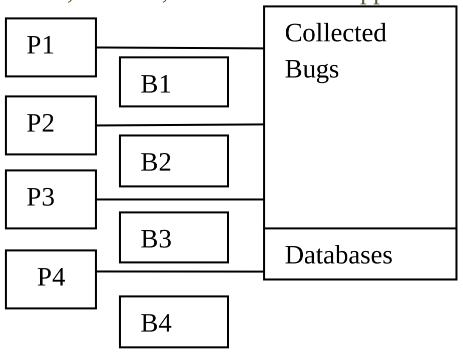

Fig 2 Bugs Collection

In the above figure 2 P1, P2, P3, P4 refers the various projects and B1, B2, B3 and B4 are the various bugs of corresponding projects. The faults are stored in any databases. These collected faults are implemented in the new application as follows. When these faults are stored we have to maintain these faults according to some predefined methods and hence it is easy to diagnose the fault in the new application. After storing, we have to check some conditions by deriving an expression. This step is very simple when using any high level languages. The expression and further operation are discussed in the following steps. (i) DETECT=FAIL COMPONENT then we confirm that the failure occurs in an application. After this step the data are recovered by means of any tool. This recovery happened instantly. When data is recovered the application is restarted. While restarting we have to ensure that the application released from failure. If failure remains establish a signal as RESTART FAILED. When the flag the application is retried and the application is running. There is a chance for new failures. For that purpose we use the signal RETRY FAILED. The retrieving of fault from the database and the remaining process are explained in the figure Fig 3. Below section gives the conducted and their results are discussed.

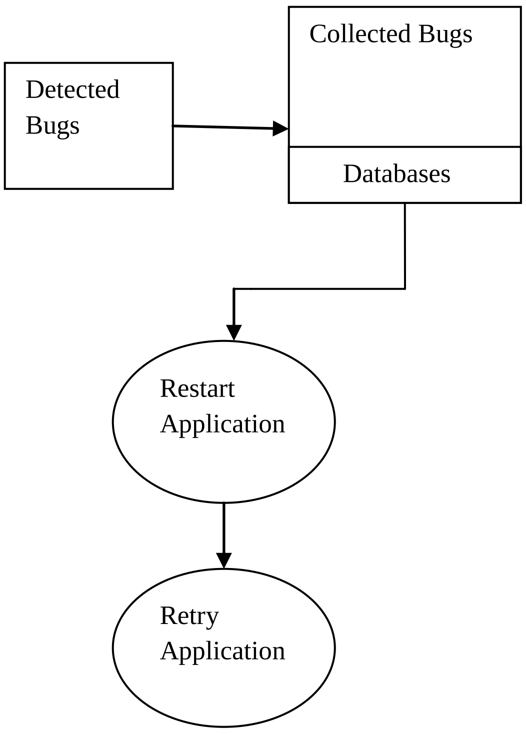

Fig 3 Retrieving bugs from database

When the detected faults and collected faults are same then restart the application after storing the results. When the data stored and the bugs are recovered then retry the same application.

IV. EXPERIMENT AND RESULTS

To evaluate the performance of this method at first we have to determine which portion of the application the measurement would be taken for reliability after that the input to the system is given. For this experiment a program in a .txt file is taken .To make the user interaction easier another .txt file is introduced where the proposed contents are stored. This work collects the bugs and hence the various .txt bugs are determined also the parameters such as KLOC and possible metrics are also collected. The bugs collected from a file which contains 10,000 LOC from various projects .The purpose of the program is to collect a data from the user in a predetermined format and implement specific tasks. According to ARBR, more than 70 bugs are collected. The bugs are determined by various projects named as P1, P2, and P3. Based on the operational profile the bugs are injected into a database. The detected bugs are also numbered and sequenced for user compatibility and maintenance. When we maintain some sequential order then it is easy to categorize the method of bugs. This method will apply to any software application which gives meaningful results. This collection appears best in all situations. Most of the models leave the development process and the results alone for consideration. In this model bugs are taken into account and so the complexity is reduced and the abstraction is achieved. The below table summarize some bugs from the considered project. The table gives the line no where the bug appeared in the project. The determined bugs are categorized by some specifications such as hardware failures, redundancy etc. The summarization of bug retrieving and their properties are given in Table 1 and the bug profile of assigned projects is discussed in Table 2.

Table 1: Bug retrieving

Bug no	Line no	Category no
2	85	3
5	536	5
7	832	7

Table 2:Bug Profile

Project	Failure	Without Failure
3	124.87	130.47
5	180.39	117.17
7	170.11	78.83

Table 3: Bug detection and repairing

Project	Detection of bug	Repaired	Without failure
3	12.862	11.592	135.520
5	9.232	8.321	124.520
7	6.550	5.321	175.321

The bug detection profile is collected from randomly taken among various projects and an average of the profile is computed for the executed applications with and without failure. These results are given in table 3 and the corresponding graph for bug profile in the new application is given in below graph Fig 4.

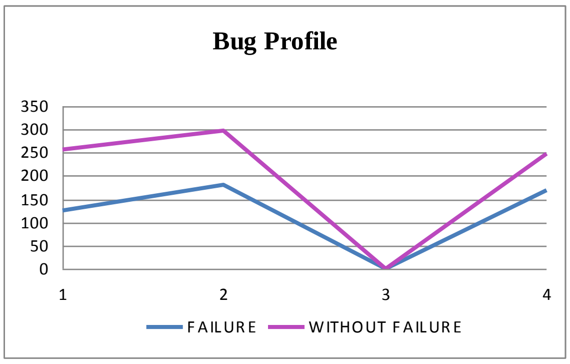

Fig 4 Bug profile analysis

The above table 3 gives the average rate of fault detection and the recovery action and repairing rate on bug process when introducing the proposed ARBR process. The above table also gives the non failure process of new application. The corresponding graph is discussed in the graph below.

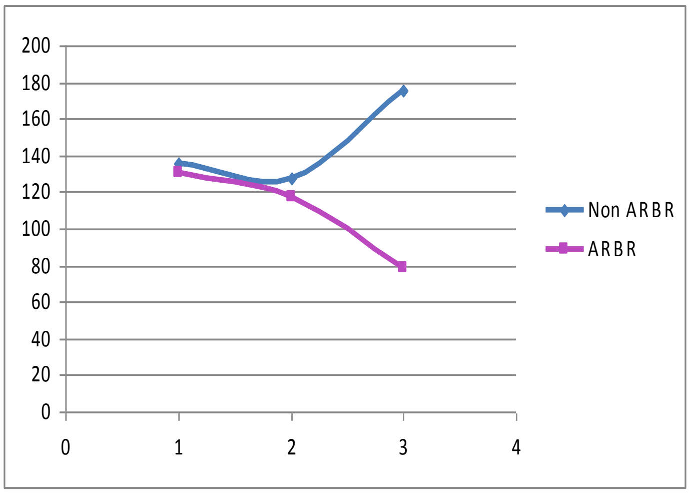

Fig 5 Reliability of new application

The above fig 5 gives that the reliability is enhanced by using ARBR. The average of bug rate is reduced considerably and hence the reliability is enhanced. By this method, 80 percent of the cases give the considerable enhancement in reliability.

This experiment shows that the proposed ARBR is implemented in the program is more efficient than existing works. The above given results indicate that the bug extraction and the retrieving are step-by-step process. But it is easier than other methods. If we complete all the steps start from extraction to retrieving successfully then ARBR is promising method to enhance reliability. This result fulfilling the objective of the work and it is a unique finding one. This method is capable to work in all environments automatically.

V. CONCLUSION

In order to achieve a required reliability, the ARBR is an optimized model. This model gives good development and it detects and repairs various failures in the application. Enhancement in reliability is hard but this model conquers the complexity. ARBR is an advanced approach but there are some limitations. When the required information's are lacking means this method is underway. The quality of ARBR depends upon the needs and quality requirements. Experiments showed that the reliability enhanced by means of ARBR and give the potential outputs without failures. It gives a promising progress to produce reliable software. Even though ARBR gives better integration between critical failure and application there is a side effect such as obtaining sensitive failures and their impact on their output. So, successive designs of ARBR for future versions will overcome these issues. This method is further enhanced as assuring defect free software and limiting of realistic constraints such as time and cost. Finally, it is concluded that the effective and the selection of a reliability model is not a simple task. The model which has low cost, time and effort is a good model. Therefore, ARBR is a good method to enhance reliability.

REFERENCES

[1] Ahmed Patel, Liu Na, Rodziah Latih, Christopher Wills, Zarina Shukur and Rabia Mulla "A Study of Mashup as a Software Application Development Technique with Examples from an End-User Programming Perspective" Journal of Computer Science ,1406-1415, 2010.
[2] Costa, E.O., A.T.R. Pozo and S.R. Vergilio, "A genetic programming approach for software reliability modeling", IEEE Trans. Reliability, pp .222-230, 2010
[3] Eduardo Oliveira Costa, Aurora Trinidad Ramirez Pozo, and Silvia Regina Vergilio A Genetic Programming Approach for Software Reliability Modeling, IEEE Transactions On Reliability, Vol. 59, no. 1, March 2010,
[4] Kapil Sharma , Rakesh Garg, Nagpal C. K, Garg R. K, " Selection of Optical software Reliability Growth Models Using a distance Based Approach" IEEE Transactions on Reliability , Vol 59, No. 2, June 2010.
[5] J. Onishi, S. Kimura, R.J.W. James, and Y. Nakagawa, "Solving the Redundancy Allocation Problem with a Mix of Components Using the Improved Surrogate Constraint Method," IEEE Trans. Reliability, Vol. 56, no. 1, pp. 94-101, Mar. 2007.
[6] Yousif A. Bastaki, "A Framework for Teaching Programming on the Internet: A Web-Based Simulation Approach" Journal of Computer Science,pp. 410-419. 2012.