International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 14 (2015) pp 34716-34719

Evaluation on software reliability approaches

Gurmanpreet Singh Sandhu[1], Rahul Kumar Singh[2]
Gurmanpreet singh sandhu
Department of computer science
Chandigarh University, Gharuan
Student
Gurmansandhusaab@gmail.com
Rahul kumar singh
Department of computer science
Chandigarh University, Gharuan
Assistant Professor
Rahulcu25@gmail.com

Abstract- In this paper two approaches are consider for measuring the software reliability which shows some effective results for each approach. Prediction model can be considered under the traditional approach. In this paper the data set is consider for measuring the reliability using the Goel-Okumoto model then the same data set is apply through the FSM(finite state machine) the results that obtained are different. Analyzing those results the FSM is better model as compared to Traditional approach.

Keywords- Software reliability, MTTF, MTTR, MTBF, finite state automata.

1. Introduction

Software reliability means failure free operation under different environments or conditions. Reliability of software shows its quality. There are some phases of software development life cycle feasibility study, design, coding, testing, implementation, maintenance by following that cycle the new product/software is developed. In this process there are some errors of different types. The probability of reliability depends upon the error. There is no software which has 100% reliability or the probability of error is zero. Human always make some mistakes which further affects the quality of software. For the quality purpose, the reliability of software should be better. It will be better if there will be less number of errors in the product/software. An important quality attribute of computer system is the degree to which it can be relied upon to perform its intended function [1].Reliability of software is important to measure because software is essentially an instrument for transforming a discrete set of inputs into discrete set of outputs [1]. Fault in software means eliminating the user requirements. Faults can come due to poor documentation or poor communication between user and programmer. The process of software testing is used to find out mistakes. Testing neither guarantees the correctness of software program. If there are errors then the reliability is low.

To estimate the reliability of the software there are some models like Jelinski and moranda , goel and okumoto model, schick and wolverton, little wood-verrall Bayesian model .Each model has advantages and disadvantages. To find accurate value of reliability need some mathematical or statistical model. Researchers want to find out accurate value of reliability.

In this paper there is technique used to determine the software reliability. The probability of failure or reliability is estimated Rahul kumar singh, Department of Computer Science, Chandigarh University,Gharuan by automata. An automaton is the best way to find out the discrete value of the software reliability. This paper consists of four sections. Section 1 explains about the software reliability and its models. In section 2 related works is explained. The basic model of reliability is explained in Section 3 along with examples. FSM is defined in Section 4. Comparison of different model is described in section 5.

2. Related Work

A. Automata-based software reliability model

R. Wason *et.al* [21] proposes Finite State Automata (FSA) based reliability model that serve as a befitting solution to all existing software reliability challenges. The model estimates actual system reliability at runtime.

Advantage**:** it allows authentic or real-time reliability estimation, prediction and can also be trained towards dynamic learning of the evolving behavior of software, and fault tolerance.

B. Software quality and reliability prediction

A. K. Pandey *et. al*[22] presents a review on software reliability and quality prediction. Software reliability and quality prediction is highly desired by the stakeholders, developers, managers, and end users. Detecting software faults early during development will definitely improve the reliability and quality in cost-effective way. One measure of software quality and reliability is the number of residual faults. A lot of models have been developed using various techniques. A common approach is followed for so1ftware reliability prediction utilizing failure data.

U. Jaglan *et. al.*[23] review the history of software reliability engineering, the current trends and existing problems, and specific difficulties.

C. Limitation and application of software reliability model

A. L. Goel[1] presents an overview and critical analysis of various modeling approaches. Paper describes several limitation and application of software reliability model during Software Development Life Cycle (SDLC).

Paper also presents step by step methodology for developing model from failure data is illustrated with example. Paper describe model user an insight into usefulness and limitation of such model that would be helpful in determining which model to use if any, in given software environment.

D. Finite automata and their decision problems

Rabin *et. al* [24] describe basic concepts of finite automata and their problems. Finite automata are considered as instruments for classifying finite tapes. Each one tape automaton defines a set of tapes; a two-tape automaton defines a set of pairs of tapes, etc. The structure of the defined sets is studied. Various generalizations of the notion of an automaton are introduced and their relation to the classical automata is determined. Some decision problems concerning automata are shown to be solvable by effective algorithms; others turn out to be unsolvable by algorithms.

3. Software Reliability Model

There are some reliability models which are used to determine the reliability of software. Basically, reliability can measure through metrics.

1. ROCOF (Rate of Occurrence of Failure): it measures the frequency of occurrence of failure.
2. MTTF (Mean Time to Failure): It is time between two successive failures.
3. MTBF (Mean time Between Failure): time between the second failures after the first failure.
4. MTTR (Mean Time to Repair): time taken to fix the failure.
5. POFOD (Probability Of Failure On Demand):it measure the likelihood of the system failing when a service request is made.

Reliability depends upon the failure occur in the program. If MTTF is known and MTTR is known then the reliability of product can be measure. Data set is given below. this is obtained by using testing tool. From testing tool the number fault is calculated and time taken to fix the failure is obtained by tester. Let's consider the data set and put the value in the equation of goel-okumoto model.

Table 1: Data set for measuring reliability

Number of test cases	Number of faults	Execution time	Time to Repair
5	0	00h:00m:05s	0s
10	5	00h:00m:15s	9s
15	2	00h:00m:21s	3s
20	0	00h:00m:34s	0s
25	0	00h:00m:45s	0s
30	1	00h:00m:55s	8s
Total=105	8	00h:02m:55S	20s

Total number of faults =8
Total number of test cases =105
Total execution time = 00h: 02m: 55S
Total time to repair = 20s
Equation of goel-okumoto model is
$\mu(t) = a(1 - e^{-bt}), a > 0, b > 0$, where
μ(t)= total actual fault, using okumoto model.
a= initially a is taken total number of fault detection
b=rate at which the defect rate decrease or increase.

From this the value of μ(t) is 8 means the reliability of the given data set is 0.08 this is depend upon the intensity function λ(t).

Procedure of using software reliability:

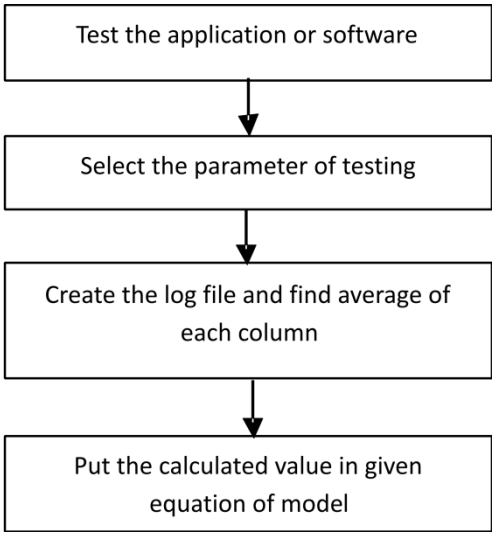

Fig: 1.Methodology for Traditional approach

From above chart the first step, test the application means collect the data in the form of code after that check the error by performing some testing process. Automated testing can be done through tools otherwise check the expected result manually.in the second step select the parameter on the behalf of testing performed.in the third step, create the log file of each test case. Last step, put the calculated values in the given equation.

4. Finite State Machine Reliability Model

Automata based software reliability estimation model that can control reliable software operation by detecting software state to state transition at runtime [2]. To implement this technique there is a figure shown below:

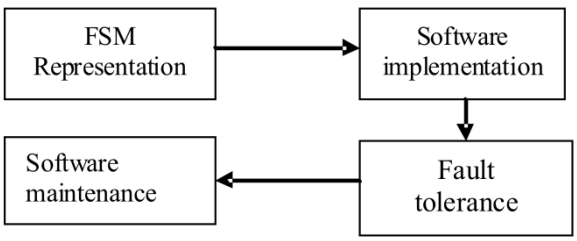

Fig. 2.Methodology for FSM
Table 2: Description table for FSM

Sr. no	Activity	Description
1	FSM Representation	Give input as code, output should be shows FSM representation and next state transition
2	Software implementation	To monitor software execution next state transition should be use.
3	Fault tolerance	Find the path using shortest path algorithm and repeat the steps.
4	Software maintenance	Record the path

In FSM reliability model there are four phases as shown in diagram. The flow starts from FSM Representation and ends at Software maintenance.

From these activities the FSM model can be represented. As the example, let's consider a code of any language, now divide the code in parts by states $q_{0,\dots,}q_n$. As the finite automata has five tuples $(Q, \sum, \delta, q_0, q_n)$

- Q is a finite set of states
- $\sum$ is a finite non empty set of inputs
- δ is transition function
- q_0 is a initial state
- q_n is a final state

Similarly FSM model will be consider these tuples in order to complete their representation. FSM representation is shown in the fig

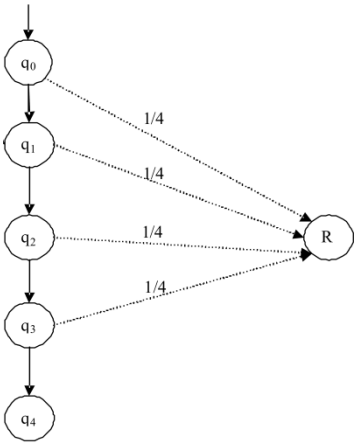

Fig. 3. Representation of software modules

Diagram conclude that each state is divide into its own reliability then total reliability R can be written as :

$R= 1/4R_1 + 1/4R_2 + 1/4R_3 + 1/4R_4$

The model cannot only allow reliability estimation and prediction but can also be trained for dynamic learning of the evolving behavior of software, and fault tolerance as it can easily retract from erroneous state to safe state during software operation [4]. The use of Finite state Machines (FSM) has also been advocated for the solution of common design problems [4]. Basically, FSM should obey two conditions:

- Each input, transition δ has to be defined.
- Defined transition has to handle cases where timeout occur.

FSM is mathematical model it gives the discrete value of the reliability. Through FSM the error detection get easier as compared to other technique. The result of each state is different due to parsing because the FSM model is provide the parsing between two stages. Hence the result will be different.

5. Comparison

The goal of estimating the reliability of a software system is to anticipate possible software failures once the software is under operation [25].FSM is better than prediction model because FSM shows discrete value of reliability. FSM an important mathematical model for software representation. Automata-based reliability prediction models have emerged as efficient models for overcoming prediction inaccuracy of traditional reliability estimation models [25].comparison between these models is just to understand the difference between the automata approach and traditional approach. If the results are compared then it clearly shows that the automata approach is better because the concept of self - learning and self - healing can be proved by automata approach. This finite state machine representation is probably a better model for software reliability estimation. In comparison the traditional reliability estimation approaches are brute-force models that use an exhaustive number of failure data with comparative ease to predict reliability of a software system [1], [25]. As we know that the prediction models are the based upon assumptions where the automata approach or FSM not based upon the assumptions. The results of both approaches are different but the better result can consider of FSM or automata approach.

The approach is already finding varied implementations in many different domains like Stochastic Petrinets and other

Petrinet-based models, weighted automata, fault trees, hierarchical state-based architectures, composite state-models etc [25].

Refrences

[1] A. L. Goel "Software Reliability Models: Assumptions, Limitations and Applicability*", IEEE Transactions on Software Engineering*,Vol. SE11, No. 12, pp. 1411-1423,1985.

[2] P.AHMED "Automata Based software Reliability Paradigm for Ubiquitous Systems", *IJCSET, ISSN*: 2229-3345,Vol.4 No. 07 July, 2013.

[3] RITIKA WASON "A Tool for runtime reliability estimation and control using automata based software reliability model", *mathematical models in engineering and computer science*, ISBN: 978-1-61804-194-4, 2013.

[4] K. Kandt, "Software Design Principles and Practices*", Jet Propulsion Laboratory*, NASA (2003).

[5] D. E. Caiazzo, J. L. Falcone, J. Hegewald, E. Lorenz, B. Stahl, D. Wang, J. Bernsdorf, B. Chopard *et. al* "A

Complex Automata approach for In-stent Restenosis: two-dimensional multiscale modeling and simulations", *Journal of Computational Science*, vol. 2, no. 1, (2011).

[6] M. E. Cambronero, G. Diaz, V. Valero and E. Martinez, "Validation and Verification of Web services choreographies by using timed automata", *The Journal of Logic and Algebraic Programming*,(2010).

[7] T. Carmely, "Using Finite State Machines to Design Software*", In EE Times: Embedded Systems Design*, vol. 23, no. 6, (2009).

[8] C. V. Ramamoorthy and F. B. Bastani, "Software reliability: Status and perspectives," *IEEE Trans. Software Eng*., vol. SE-8, July 1982.

[9] B.M. Gouthami and P.Kumar "Effective Path Selection to Estimate Software Reliability", *Special Issue of International Journal of Computer Applications*, ICCCMIT, 2012.

[10] M. Satyanarayanan "Pervasive Computing: Vision and Challenges", *IEEE Personal Communications*, Vol. 8, No. 4, Aug. 2001.

[11] J. L. Cook and J. E. R. Marquez, "Reliability Analysis of Cluster-based Adhoc Networks", *Reliability Engineering and System Safety*, vol. 93, no. 10, (2010).

[12] M. Crochemore and D. M. Gabbay, "Reactive Automata, Information and Computation", vol. 209, no. 4, (2011).

[13] I. K. El-Far and J. A. Whittaker, "Model-based Software Testing", *Encyclopedia on Software Engineering*, John Wiley & Sons, Inc, (2002).

[14] D. Ghosh, R. Sharman, H. R. Rao and S. Upadhyaya, "Self-Healing Systems- Survey and Synthesis", *Decision Support Systems*, vol. 42, no. 4, (2007).

[15] S. S. Gokhale, "Accurate Reliability Prediction Based on Software Structure", http:// www.engr.uconn.edu/~ssg/ cse300/ 397-232.pdf, (2004).

[16] N. Langberg and N. D. Singpurwalla, "Unification of some software reliability models via the Bayesian approach,*" SIAM J. Sci. Statist. Computer*, vol. 6., pp. 781-790, July 1985.

[17] P. Zhang, H. Muccini and B. Li, "A Classification and Comparison of Model Checking Software Architecture Techniques", *The Journal of Systems and Software*, vol. 83, no. 5, (2010).

[18] D. J. Smith, "Reliability, Maintainability and Risk: Practical Methods for Engineers", Butterworth-Heinemann, (2001).

[19] J. Strassner, "An Overview of the Challenges and Promises of Autonomic Computing", *CSI Communications*, (2011).

[20] R. H. Reussner, H. W. Schimdt and I. H. Poernomo, "Reliability -Prediction for Component-based Software Architectures", *The Journal of Systems and Software*, vol. 66, no. 3, (2003).

[21] R. Wason, P. Ahmed and M. Qasim Rafiq "Automata-Based Software Reliability Model: The Key to Reliable Software", *IEEE transaction of Software Engineering and Its Applications* (2011).

[22] A. K. Pandey and N. K. Goyal "Background: Software Quality and Reliability Prediction", *Early Software Reliability Prediction, Studies in Fuzziness and Soft Computing* 303, DOI: 10.1007/978-81-322-1176-1_2, Springer India 2013.

[23] U. Jaglan, D. Shrivastava "A straight map of Software Reliability on component based Engineering", *International Journal of Advanced Research in Computer Science and Software Engineering* Volume 2, Issue 3, March 2012.

[24] Rabin and D.Scott "Finite Automata and Their Decision Problem", *IBM Journal*.

[25] R. Wason, P. Ahmed and M. Q. Rafiq" Automata-Based Software Reliability Model: The Key to Reliable software" *International Journal of Software Engineering and Its Applications* Vol.7, No.6 (2013), pp.111-126, 2013.