Design of a Reliability Model under Different Fault based Parameters

Yukti Mehta
Student, M.Tech
ITM University
Gurgaon-122001

Aman Jatain
Assistant Professor, CSE & IT Department
ITM University
Gurgaon-122001

ABSTRACT
Software fault analysis is on major vector adopted by different researchers to analyze the software reliability. But most of the author taken the fault individually as the fault criticality or the fault count. But in this presented work we have considered all aspects of software faults i.e. fault count, fault criticality, fault frequency, associatively between faults. In this work a three level structure is been defined to perform the fault based analysis. At first level, individual fault and fault criticality is analyzed where as in second level fault-module associatively and fault-fault associatively is discussed. At third level, the complexity of the module will be identified. Finally, all factors will be combined to identify the overall complexity and severity of the module and the system. As the work is based on all aspects of faults as well as metric based complexity, so that high level reliability and accuracy is expected from the system.

Keywords
Software Fault Criticality; Fault-Fault Associatively; Fault-Module Associatively

1. INTRODUCTION
Software development process is not only about to deliver software but also to deliver a quality product. There are number of different aspects that represent the software quality according to the view point of different stakeholders. But the main consideration is given to the end user for which the software product is developed. The reliability vector is the foremost requirement for end user to accept the product as a quality product. Software reliability itself is not the single characteristics; it is itself a vast term defined under different parameters. The key terms associated with the software reliability is the software fault. Software fault itself is a wide term defined under different aspects. It can be as small as a warning or it can be as critical as a software failure. There are number of number of existing models that work under the software fault analysis [1] [2]. Some of the factors considered by different reliability models under fault consideration are shown in figure 1.

The fault criticality is defined as the type of fault considered by for a software project. Fault criticality can be categorized as a bug, error or the failure. The bug is the fault type that occurs in a software system but there is no such sequence or the rule about the generation of the bug again and again. Bug is not occurred frequently that can occur because of some external reference such as memory requirement, platform error, dependency issue etc. The error is the software fault that occurs frequently in a software module. As the module will be executed, the fault in that software module disturbs the execution flow by breaking the execution process or some process drop. The most critical fault type is the software failure, as the failure occur, the complete system stop working and the software crash can occur. This kind of fault is more destructive [3] [4].

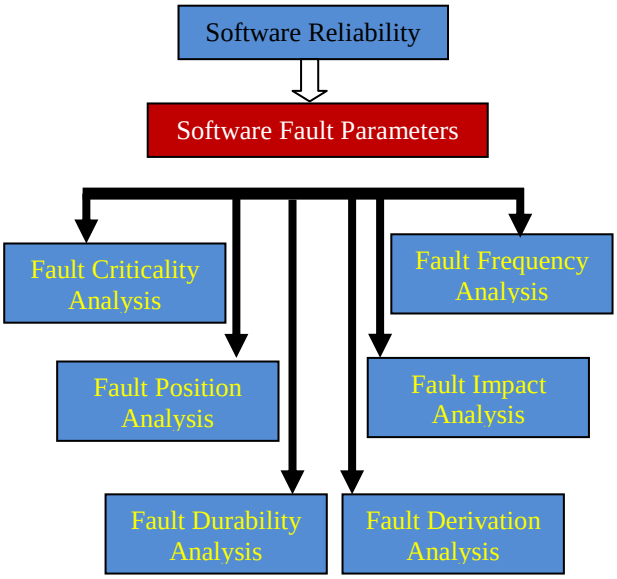

Figure 1. Software Reliability Constraints

Another consideration associated with software fault analysis is the fault frequency. The frequency defines the number of times the fault occurs in a particular time interval. Fault frequency is also the parameter to decide the fault criticality. A fault or error becomes the failure, if the frequency of the software fault increases. Another associated feature of software fault is the fault position analysis. When the fault occur in the execution process, in such case the position of fault occurrence also matter. If the fault position is in the beginning, the fault criticality is less whereas if the fault occurs in the later stages, the criticality of the fault is higher [6] [7].

Fault Module analysis is another vector that defines the fault criticality. The faulty module is having the higher importance while deciding the software reliability. Software Impact Analysis is another vector that basically defines the consequences when a software fault occurs in a software system. If the fault gives an error message but process normally, it can be ignored and less critical but if the error gives the system restart or the data loss it is more critical. Durability is another vector defined with fault analysis. Durability is about to identify the existence time of the error in a software system. Higher the time, the error stays in the system, more critical the error will be. The last consideration is about to identify the fault derivation analysis. It means is

the fault itself generating some other faults over the system or not. If it generates more faults, the software fault is most critical to that [8] [9].

From this study, we identified all the fault related vectors that affect the software reliability. The main consideration for the fault based reliability estimation is the quantity the software fault in the software system or in a particular software module. The fault estimation function is shown in figure 2.

1.1 Fault Function

The failures in a software system can be defined and determined by using different function. Some of these functions include the failure intensity function, failure rate function, failure intensity function etc. These functions basically provide the quantitative values to represent the software module or software fault or software fault criticality [10] [11]. The cumulative failure function basically represents the expected cumulative failure at particular instance of time. It is also known as mean-value function. Another function to estimate the software fault is the failure intensity function. This function is based on the cumulative failure function. It estimate the software criticality based on the variation analysis on cumulative failure function. Another function for failure analysis is the failure rate analysis function. It is the probabilistic function that identifies the number of failure in specific time interval. The number of failures between two time slots is represented by failure intensity function [12] [13].

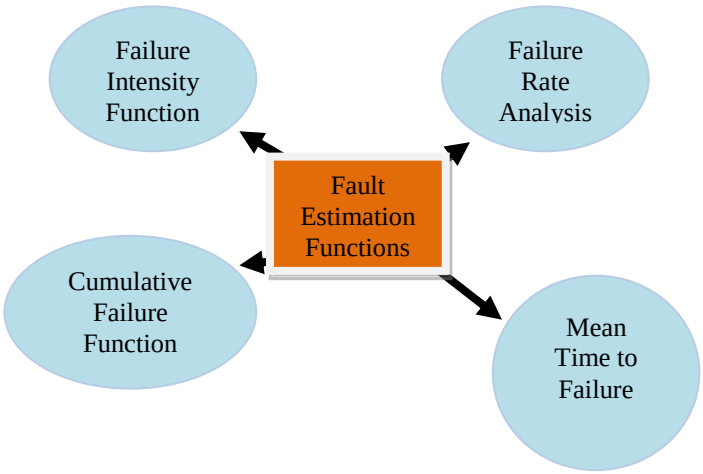

Figure 2. Failure Estimation Functions

Another effective failure estimation function is mean time to failure (MTTF). It gives the estimation of next time instance when the failure can occur over the system. Similar to this, Mean Time to Repair (MTTR) is the function that estimates the time instance of system repair if the failure over the system is identified. Based on these two vectors, the probabilistic estimation can be done to identify the system availability once the repairing of fault is done. The availability is represented as

MTTF Availability = MTTF / MTTF + MTTR

In this work, a failure estimation based work is been defined to analyze the software reliability. In section I, the introduction software fault and all fault related vectors are discussed. In section II, the work performed by different authors is discussed. In section III, the proposed work is defined along with algorithmic specification. In section IV, the results obtained from the work are discussed. In section V, the conclusion derived from the work is discussed.

1.2 Comparison on Different Reliability Models

There are number of reliability that works on different parameters and different methodologies to perform the software analysis. In this section, the comparison is been defined between three main reliability models. These models are Software Metrics Model, Fault Based Model and Musa Model. The comparative analysis under different parameters is shown in table 1

Table 1. Comparison on Reliability Models

Parameters	Software Metrics Model	Fault Based Model	Musa Model
Known As	Software Evaluation Model	Weibull Failure Model	Execution Time Model
Description	It contains a set of process and product Metrics for Software Estimation	Perform analysis on software fault, fault frequency and failure analysis	It performs the execution time analysis and interval time analysis between failures
Example	SLOC, Reusability Analysis, Portability Analysis	Lifetime Analysis, Fault Count Analysis, Failure Analysis	Elapsed Time analysis between software failure and the actual calendar time

2. EXISTING WORK

Lot of work is already done by different authors to perform the risk analysis and the software quality analysis. Some of the work done by earlier authors, under different reliability vectors and on different reliability models, is discussed in this section.

Liguo Huang has defined a risk assurance based work for a text mining oriented software system. Author has defined the methodology to establish the methods and aim under the risk analysis and the risk associations. Author identifies the frequency analysis under different vectors for historical projects. Author also performed the analysis on e-service projects and presented the risk associated approaches so that the effective software development will be done in real applications [1]. Mary Sumner has defined an enterprise wide information management system under different risk factors. These management projects are analyzed on different data management software like SAP, Oracle etc. Author also performed the history based analysis so that the effective software analysis will be derived [2]. Andreas Schmietendorf presented a process model so that the performance analysis on the engineering tasks will be performed. It also includes the investigation and evaluation so that the software development and performance model is defined in this work. Author defined the task oriented analysis model under the quantitative framework in which the identification of requirement and the resources is been performed under the risk analysis and the performance information analysis [3].

In Year 2005, Guillaume Langelier has defined a visualization approach for the quality estimation on large software projects. Author defined the complex software project analysis under the development and maintenance approaches so that risk will be minimized. Author has defined the work in an effective opportunistic risk analysis approach so that the software reliability over the system will be achieved. Author defined the for open source programs. The presented framework to the system is been defined under quality modelling so that the large scale software system will be defined [4]. Another work on risk management was proposed by Mira Kajko-Mattsson. Author defined the software risk management responsibilities and to represent the software risk management responsibilities. It also includes the software risk analysis approaches at different phases of software system [5]. In Year 2006, Ossi Taipale has defined an observation based approach to improve the software system. Software defined a qualitative as the complex practice so that knowledge based process study will be performed. The work also includes the testing cost analysis and its relation with software quality. It includes the survey on testing on the organization units that are interviewed by the author. Author defined a study on the research method so that the theme based interviews were performed [6].

In Year 2012, Michael Grace has defined the risk ranking for a software system so that the estimation of software system will be done under the accuracy and the scalability analysis. Author defined the work for android based applications. Author defined a sampled risk analysis under the security vector. The work was testing on real time environment for trusted and untrusted applications [7]. In Year 2010, Adailton Mag alhaes Lima defined under a simulation environment so that the project analysis for the data and other risk factors will be analyzed. Author defined the probabilistic analysis for the project management and the risk assessment. The development process along include the decision making so that improvement software management will be done [8].

3. RESEARCH METHODOLOGY

In this present work, a statistical parametric analysis approach is been presented to perform the software risk estimation. The presented approach is defined under three fault based vectors. In the first stage, the fault analysis over the software system is identified and analyzed. In this analysis state, the prioritization of the software fault is done. The fault identification is actually the fault association respective to the modules is identified. The complete software system is dived in terms of software modules and each module is defined along with integrated software faults. After the fault identification, the software fault prioritization is done shown on figure 3. The prioritization can be of based on fault frequency or based or on fault criticality.

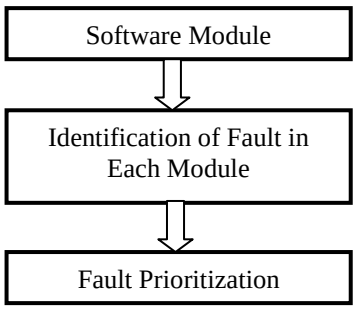

Figure 3. Software Fault Analysis

Once the fault priorities are identified, the next work is to find the association between software faults. The association between the software faults is called software fault dependency.

The independent faults are comparative less effective than dependent faults. The fuzzy rule is suggested here to perform the cost estimation on each software fault and the fuzzy operators are applied to identify the association between these faults. Based on this, the criticality of each software module will be identified individually. The cost estimation process is shown in figure 4.

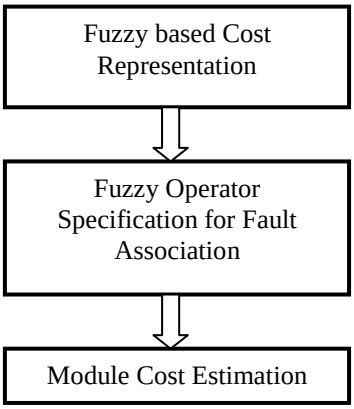

Figure 4. Module Cost Estimation

At the final stage, the estimation of the software cost will be done by performing an aggregative cost analysis. While estimating the aggregative cost, at first the weight age is assigned to different software modules. Based on this weight age assignment, and fault individual module cost estimation, the impact of the module will be identified. The aggregation on these vectors will be performed to identify the overall software cost or the software cost criticality.

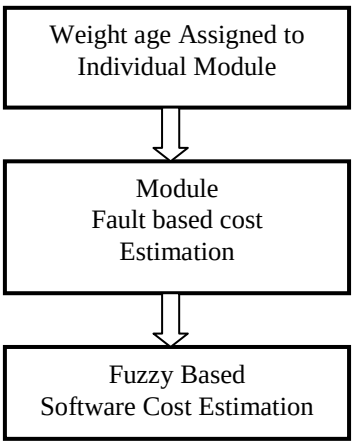

Figure 5. Software Cost Estimation

The presented work is shown in three different stages and each stage is defined as a smaller process model. The process models collectively forms the fault analysis and the cost analysis. The fuzzy based provide the effective results from the system.

4. RESULTS

The presented work is implemented in Matlab environment. The work is performed on a software system that is divided in N modules logically. Each software module is having the

importance based on the distance from the goal state. With each software module, software faults will be identified. The estimation of the fault vector in each software module is done under the defined model.

The results obtained from the proposed aggregative fuzzy based system are shown as under. The analysis is here been done under the module based fault criticality analysis. Here figure 6 is showing the results obtained at high criticality level. Identification of modules under different criticality levels.

Here figure 6 is showing the criticality analysis of different modules. Here, the criticality of each module is represented between 0 and 1. The closer the criticality value to 1 is more faults critical the module will be. As shown in the figure module 3 is most critical module and modules 1, 4 and 7 are least critical.

5. CONCLUSION

The presented work is about to perform a software reliability estimation under fault analysis. In this work, two levels fuzzy logic is defined to perform the criticality analysis as well as cost analysis. The first level is implemented on individual modules and second level is implemented on aggregative cost. The obtained results show the clear module of criticality under fault vector.

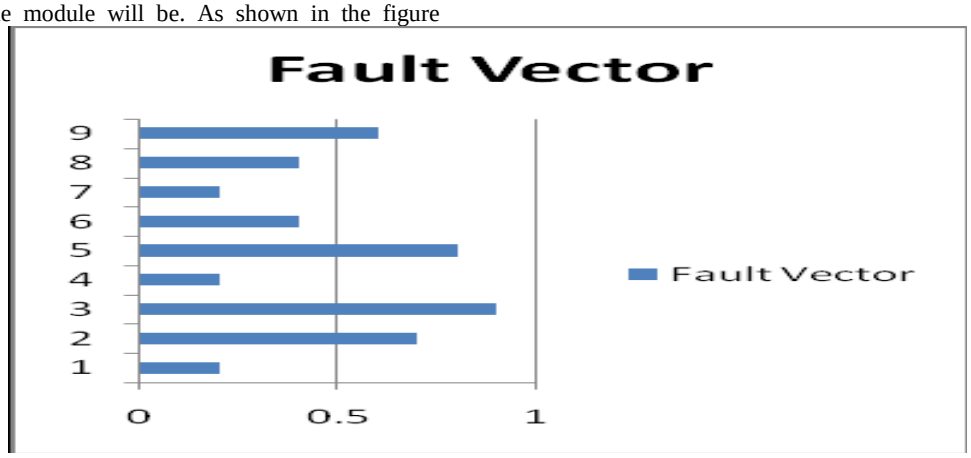

Figure 6. Module Criticality Analysis

6. ACKNOLEDGMENTS

This research is supported by the Department of Computer Science and Information Technology at ITM University.

7. REFERENCES

[1] LiGuo Huang, "Text Mining in Supporting Software Systems Risk Assurance," ASE'10, September 20-24, 2010, Antwerp, Belgium. ACM 978-1-4503-0116-9/10/09.

[2] Mary Sumner, "Risk Factors in Enterprise Wide Information Management Systems Projects," SIGCPR 2000 Evanston Illinois USA 1 -58113-212-x/00/04.

[3] Andreas Schmietendorf, "Process models for the software development and performance engineering tasks," WOSP '02, July 24-26, 2002 Rome, Italy ACM ISBN 1-1-58113-563-7 02/07.

[4] Guillaume Langelier, "Visualization based Analysis of Quality for Large scale Software Systems," ASE'05, November 7–11, 2005, Long Beach, California, USA. ACM1-58113-993-4/05/0011.

[5] Mira Kajko-Mattsson, "Laying out the Scope of Developers' Risk Management Responsibilities," ICIS 2009, November 24-26, 2009 Seoul, Korea ACM 978-1-60558-710-3/09/11.

[6] Ossi Taipale, "Improving Software Testing by Observing Practice," ISESE'06, September 21–22,2006,Riode Janeiro, Brazil. ACM 1-59593-218-6/06/0009.

[7] Michael Grace, "RiskRanker: Scalable and Accurate Zero-day Android Malware Detection," MobiSys'12, June 25–29, 2012, Low Wood Bay, Lake District, UK. ACM 978-1-4503-1301-8/12/06.

[8] Adailton Mag alhaes Lima, "Risk Assessment on Distributed Software Projects," ICSE '10, May 2-8, 2010, Cape Town, South Africa ACM 978-1-60558-719-6/10/05.

[9] Margaret-Anne Storey, "The Impact of Social Media on Software Engineering Practices and Tools," FoSER 2010, November 7–8, 2010, Santa Fe, New Mexico, USA. ACM 978-1-4503-0427-6/10/11 .

[10] Peter Hearty, "Automated Population of Causal Models for Improved Software Risk Assessment," ASE'05, November 7–11, 2005, Long Beach, California, USA. ACM 1-58113-993-4/05/0011.

[11] C. R. Rene Robin, "Development of Educational Ontology for Software Risk Analysis," ICCCS'11, February 12–14, 2011, Rourkela, Odisha, India. ACM 978-1-4503-0464-1/11/02.

[12] Lucas Layman, "A Case Study of Measuring Process Risk for Early Insights into Software Safety," ICSE'11, May 21–28, 2011, Waikiki, Honolulu, HI, USA ACM 978-1-4503-0445-0/11/05.

[13] C R Rene Robin, "An Ontology Based Linguistic Infrastructure to Represent Software Risk Identification Knowledge," ICWET'11, February 25–26, 2011, Mumbai, Maharashtra, India. ACM 978-1-4503-0449-8/11/02.