A Fuzzy Based Analytical Model to Estimate Software Reliability under Fault Analysis

Loveleen Kaur¹, Mohit Kumar²
¹Student, M. Tech., RIMT-IET, Mandi Gobindgarh (Punjab) loveleen_bilnn@yahoo.com
²Assistant Professor, RIMT-IET, Mandi Gobindgarh (Punjab) mohit.nabha@gmail.com

Abstract
Software Reliability is one of the most crucial vectors to estimate the software quality. There are number of possible directions to analyze the Software Reliability. One of such vectors is Software Fault analysis. In this present work, the discussion has been performed on the software fault under different aspects. Fault is here defined in association with software modules. These faults are been described under different criticality level as well as their associated dependency while resolving the fault. The fuzzy based model is been presented to estimate the software reliability. The obtained results show the clear separation of the software modules under different criticality levels.

Keywords: Software Criticality, Fault Analysis, Fuzzy based Model, Fault Association.

1. Introduction

When we talk about the software quality, the foremost thing is considered is the reliability. The reliability itself is directly related with software faults and failures. Higher the probability of software failures in a software system, less reliable the software system will be. Each product, either it is hardware or the software is been analyzed under its worst performance vector. Fault based analysis or testing is the major concept to identify the criticality of such system. In software market the fault based testing play an important role to identify the software system reliability. Software Reliability has been measured under different vectors including the

- Maximum and Average Load on system.
- Execution Time Analysis
- Platform Dependency
- Stability Vector
- Software Fault Analysis etc.

These all vectors are important but these all vectors are been analyzed under the fault aspect. Software fault itself is been defined under different parameters as shown in figure 1.

![Figure 1: Vectors Associated with Software Fault](image_url)
There are different models associated with software reliability. When a software fault is so crucial to the software system and stop the basic working of software system or give some data loss such software fault is considered as the failure. Software failures required immediate action as we cannot process any section of software system with some failure in it. Another important aspect associated with software system is the “Effect of Fault”. If the software fault just show some error message or warning and does not give any data loss then it is less critical. If the software fault stops the system processing and restarts the activity or the software system, the criticality is higher. If the software fault gives the data loss and disturbs other applications or associated software also then it is most critical software fault. One more important aspect associated with software fault is “Criticality of Associated Module”. If the fault is critical but the module with which it is associated is less critical, the criticality of fault also decreased. Such as some fault occur in help module or the review modules are less critical then the fault occur in backup or recovery modules. The frequency of execution of such software module is also an important factor while analyzing the software faults.

The presented work defines the software reliability estimation under the fault analysis under different vector. The vector considered in this work are the fault criticality, association with software module and the dependency between the software faults. In this section we have described the importance of software faults under the criticality vector, in section II, the work done in same area is been discussed. In section III the presented fuzzy based model is been described and in section IV, the result driven from the system are presented. In section V, the conclusion driven from this research work is presented.

2. Reliability Models

There are different models associated with software reliability to present different aspects of software quality estimation under software fault analysis. Most of these models are based on software fault and frequency based, but some models also considered the software metrics as the major concern. In this section some of these models presented by earlier researchers. Some of these models are adopted by different organizations. In this section, some of these models are described in detail. One of such model is “Weibull Failure Model”. This is the analytical model based on the estimation of software fault in the software system. This model incorporates the software defect analysis with software reliability. This model is basically the estimation model to perform the fault based analysis over the system. This reliability model is been defined based on the fatigue value analysis under the defect constraint. This model is been used for software as well as hardware based system. This model includes the process modeling under the fault and failure based estimations. This model is helpful to identify the average and actual life of a software system. This model is very useful in real time systems that can be a software as well as hardware systems.

The another model presented based on the execution time analysis is the “Musa Model”. This model can predict the software system based on the process time and execution time of the software systems. The prediction of software reliability is been defined under the development phase and some decision vectors are also applied so that the software design analysis is also been performed. This model analyze the system under the time constraint and based on pass on distribution as the basic distribution of software reliability over the system. The limitation of this model is about the assumption that the software system will not introduce new faults or the failure. The system is effective for the known faults and its estimation. These assumptions are based on model generation as well estimation of software faults in upgraded codes so that the generated system will be bug free. In addition to limitations by assumptions, Musa’s Basic Execution Time Model is based on the statistical model so that it includes some undesirable biases and gives the estimation at low confidence level.

The another model considered by many authors is the “Monte Carlo Model”. This model is introduced in 1940 as the part of atomic bomb program. It is been not only provide the software reliability estimation but also defined estimation for different projects associated with physics and finance stream. This system includes the process modeling under the uncertainty analysis. This model is defined as a set of random test that are performed on running system and estimate the software reliability. This model is effective for the complex problems such as in case of brute force attack. This model is expensive in terms of time and cost. The reliability estimation by this model is done effectively.

3. Existing Work

In Year 2010, Margaret-Anne Storey performed a work,” The Impact of Social Media on Software Engineering Practices and Tools”. In this position paper, Author advocate for research that strives to understand the benefits, risks and limitations of using social media in software development at the team, project and community levels. Guided by the implications of current tools and social media features, Author propose a set of pertinent research questions around community involvement, project coordination and management, as well as individual software development activities. Answers to these questions will guide future software engineering tool innovations and software development team practices. In Year 2005, Peter Hearty performed a work,” Automated Population of Causal Models for Improved Software Risk Assessment”. Recent
work in applying causal modeling (Bayesian networks) to software engineering has resulted in improved decision support systems for software project managers. Once the causal models are built there are commercial tools that can run them. However, data to populate the models is typically entered manually and this is an impediment to their more widespread use. In Year 2011, C. R. Rene Robin performed a work, "A Case Study of Measuring Process Risk for Early Insights into Software Safety". In this case study, Author examine software safety risk in three flight hardware systems in NASA's Constellation spaceflight program. Author applied Presented Technical and Process Risk Measurement (TPRM) methodology to the Constellation hazard analysis process to quantify the technical and process risks involving software safety in the early design phase of these projects.

In Year 2011, C R Rene Robin performed a work," An Ontology Based Linguistic Infrastructure to Represent Software Risk Identification Knowledge". This paper describes the processes of conceptualization and specification, or building of, an ontology. The domain for which the ontology has been constructed is software risk identification. The required concepts, the semantic description of the concepts and the interrelationship among the concepts along with all other ontological components have been collected from various literatures and experience of the people from software industry. In Year 2006, Wiboon Jiamthubthugsin performed a work," Portfolio Management of Software Development Projects Using COCOMO II". This paper proposes integrating portfolio management with COCOMO II that offers more management flexibility. Managers can adjust other resources, such as tools, staff capability, communication support, etc. to improve the project's success. The proposed method can also be applied despite limited historical data and expert judgment. In addition, this paper introduces time constraints into portfolio management without assuming unrealistic linearity between effort and time.

In Year 2006, Charles X. Ling performed a work," Maximum Profit Mining and Its Application in Software Development". In this paper, Author develop an Escalation Prediction (EP) system that mines historic defect report data and predict the escalation risk of the defects for maximum net profit. More specifically, Author first describe a simple and general framework to convert the maximum net profit problem to cost-sensitive learning. In Year 2011, Julius Davies performed a work," Measuring Subversions: Security and Legal Risk in Reused Software Artifacts". In Presented exploratory study Author propose a manual technique to locate documented security and legal problems in a set of reused software artifacts. Author evaluate Presented technique with a case study of 81 Java libraries found in a proprietary e-commerce web application. Using Presented approach Author discovered both a potential legal problem with one library, and a second library that was affected by a known security vulnerability.

4. Research Methodology

In this present work a software reliability based model is been presented to analyze the software under the fault vector. The fault itself is here defined under three main vectors. First vector considered in this work is the fault criticality. The fault criticality is based on the type of fault as well as its impact on the software system. Once the faults are been categorized, the next work is to defined the software modules along with associated faults. The complete software system is been divided in terms of software modules and each software module is been defined along with associated software faults. Once the associated software modules and software faults are been described, the next stage is to describe the association between software faults. The association between two software fault faults is defined in terms of dependent and independent faults. The fault association with the software system is described as under

<table>
<thead>
<tr>
<th>Software Faults</th>
<th>Association</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 &amp; F2</td>
<td>True Dependent</td>
</tr>
<tr>
<td>F1</td>
<td>F2</td>
</tr>
<tr>
<td>F1</td>
<td>Independent</td>
</tr>
<tr>
<td>F2</td>
<td>Independent</td>
</tr>
</tbody>
</table>

Here table 1 is showing the dependency between the fault faults. If one fault occur only after the occurrence of other fault it is called true dependency. Whereas if two faults exist in such way, if one will exist then other will not such dependency is called Partial Dependency. If occurrence of one fault does not affect the other, then these faults are called independent faults. The basic flow of presented approach is shown in figure 2.
A) Fuzzy Logic

The concept of fuzzy logic (FL) is not a control methodology, but it is a way of processing data by allowing partial set membership rather than crisp set membership or non-membership. Professor Zadeh reasoned that people do not require precise, numerical information input, and yet they are capable of highly adaptive control. If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement. Fuzzy Logic is here been applied to identify the software modules based on the criticality level, we have divided the criticality of the software module under three levels called High, Medium and Low. A software module having multiple software defects can be less critical even if some module is having one high critical fault can be more critical. Under this aggregative analysis the software criticality is been estimated and based on this estimation software reliability over each module is defined and categorized.

5. Results

The presented work is been implemented in Matlab 7.8 Environment. To present the work we have taken a dummy dataset with 10 Software modules and the associated modules are defined along with modules, These modules are listed in table 2.

Table 2: Fault Associated Modules

<table>
<thead>
<tr>
<th>Module ID</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1</td>
<td>F1, F2, F3</td>
</tr>
<tr>
<td>Module 2</td>
<td>F1, F4, F5</td>
</tr>
<tr>
<td>Module 3</td>
<td>F1, F2, F4, F6</td>
</tr>
<tr>
<td>Module 4</td>
<td>F1, F2, F8</td>
</tr>
<tr>
<td>Module 5</td>
<td>F1, F9, F10</td>
</tr>
<tr>
<td>Module 6</td>
<td>F1, F2</td>
</tr>
<tr>
<td>Module 7</td>
<td>F1, F5, F7</td>
</tr>
<tr>
<td>Module 8</td>
<td>F1, F3, F8</td>
</tr>
<tr>
<td>Module 9</td>
<td>F1, F2, F4</td>
</tr>
<tr>
<td>Module 10</td>
<td>F1, F8</td>
</tr>
</tbody>
</table>

The results obtained from the system are been presented in terms of identification of modules under different criticality levels. Here figure 3 is showing the highly critical modules.
represents the criticality level under fuzzy based fault analysis. As we can most of the modules in software system are highly critical so that overall system is critical.

Figure 4: Less Reliable Modules

Here figure 4 is showing the less reliable modules. As we can, all the modules that are not critical are even not much reliable so that we can conclude; all the modules in the software system are either having the criticality level more than average.

6. Conclusion

Software fault analysis is one of the major concerns to estimate the software reliability. In this paper, the software faults are analyzed in association with other faults as well as in association with software modules. In this work, a fuzzy based model is been defined to identify the critical modules and based on this analysis, the criticality of software system is been analyzed. The obtained results show the clear identification of critical modules in the software system.

References


