A Fuzzy Based Fault Analysis Approach to Analyze Software Quality

Garima Chawla* and Santosh Kr. Thakur

*CSE Department, BRCM College of Engg. &Tech, Bahal, Haryana

Accepted 02 August 2013, Available online 14 August 2013, Vol.3, No.3 (August 2013)

Abstract

One of the important aspects of software development is the software quality. There are different measures of software quality analyzed by different researchers. One of such measure of software quality is the software fault analysis. Software fault analysis itself defines the criticality of software system. There are different parameter to perform this risk analysis. The presented work is about the study of software reliability under risk model. In this paper, a fuzzy based software fault analysis system is defined to analyze the modular quality of software system. At the initial stage, the modules are defined with associated faults and their criticality. In second stage, the relationship between these faults is presented to estimate the module criticality and at the final stage the software quality is defined for the whole system. The work is implemented in matlab environment and obtained results show the effective estimation of software quality under defined parameter and approach.

Keywords: Software Quality, Fuzzy System, Software Risk, Fault Analysis

1. Introduction

Software fault is one of the major factors to decide the software cost, reliability and the software deadlines. As the software proposal is presented along with this fault analysis also begin. With each step of software development some risk factor is added with software cost. Other than this some other factors like availability of resources, software duration also affect the software risk. In this present work will evaluate the software aggressive risk based on some of these factors. The presented work is about to evaluate the software fault analysis based on some weightage. The assessment of software quality will be done using some soft computing techniques. Here the risk is divided in different categorize based on the risk impact and then an risk aggregation based computation is implemented to find actual risk in the software system.

Hardware Reliability: - It is the ability of hardware to perform its functions for some period of time without any failure. It is expressed as MTBF which means mean time between failures. The hardware reliability is described on the basis of bath tub curve

Software Reliability: - Software Reliability is the probability of failure free software which work for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability. Software reliability is different from hardware reliability because it defines the design perfection, rather than manufacturing perfection. Below diagram describes the software reliability.

2. Review of literature

Shyi-Ming Chen performed a work, Fuzzy group decision making for evaluating the rate of aggregative risk in software development. In this paper, Author present a new algorithm to evaluate the rate of aggregative risk in software development using fuzzy set theory under the fuzzy group decision making environment. The proposed algorithm has the following advantages: (1) It does not need to form the fuzzy assessment matrices for attributes. (2) It does not need to perform the complicated defuzzification operations of fuzzy numbers using the centroid method. Huey-Ming Lee performed a work. A new algorithm for applying fuzzy set theory to evaluate the rate of aggregative risk in software development. In this paper, Author presents a new algorithm to tackle the rate of aggregative risk in fuzzy circumstances by fuzzy sets theory during any phase of the life cycle. The proposed algorithm is easier, more flexible and useful than the ones they have presented before. Huey-Ming Lee performed a work. Applying fuzzy set theory to evaluate the rate of aggregative risk in software development. The purpose of this study is not only to build a structure model of risk in software development but also evaluate the rate of aggregative risk by fuzzy set theory. While evaluating the rate of aggregative risk, Author may adjust the weights or grades of the factors until Author can accept it. Author also shows that the rate of aggregative risk is reasonable.

*Corresponding author Garima Chawla is a CSE student and Santosh Kr. Thakur is working as Asst. Prof.
Huey-Ming Lee performed a work, Fuzzy Presumptive Rate of Aggregative Risk in Software Development. In this paper, Author proposes the computational rule inferences to tackle the presumptive rate of aggregative risk in fuzzy circumstances. The proposed method of presumptive rate of aggregative risk directly uses the fuzzy numbers rather than the linguistic values to presume. The presumptive rate is close to the human thinking. John P. Hudepohl performed a work, Integrating Metrics and Models for Software Risk Assessment. This paper shows how a system such as EMERALD can enhance software development, testing, and maintenance by integration of (1) a software quality improvement strategy, (2) measurements and models, and (3) delivery of results to the desktop of developers in a timely manner. This paper also summarizes empirical experiments with EMERALD’S models using data from large industrial telecommunications software systems. Tim Menzies performed a work, Practical Large Scale What-if Queries: Case Studies with Software Risk Assessment. Author describe here a general method for understanding that data. Large scale what-if queries can guide Monte Carlo simulations of a model. Machine learning can then be used to summarize the output. The summarization is an ensemble of decision trees. The TARZAN system can poll the ensemble looking for majority conclusions regarding what factors change the classifications of the data. Say-Wei Foo performed a work, SOFTWARE RISK ASSESSMENT MODEL. In this paper, a new model for the assessment of risk in software projects is proposed. The model, named Software Risk Assessment Model (SUM), makes use of a comprehensive questionnaire. Test results show that using the risk indicator obtained from the SRAM, it is possible to predict the possible outcome of software projects with good accuracy. Mikhail Auguston performed a work, Using Attributed Event Grammar Environment Models for Automated Test Generation and Software Risk Assessment of System-of-Systems. This paper presents some concepts, principles, and techniques for automated testing of system-of-systems based on attributed event grammar (AEG) modeling of the system’s operational environment. AEG provides a uniform approach for automatically generating, executing, and analyzing tests. Quantitative and qualitative software risk assessment can be performed based on statistics gathered during automatic test execution within the specified environment model. Rattikorn Hewett performed a work, Assessment of Software Risks with Model-Based Reasoning. This paper presents a model-based reasoning approach for assessment of software risks by employing knowledge models of software faults and the system in which the software is used along with reasoning mechanisms that allow automated qualitative inferences for hazard analysis. Author gives an example of a software controller for a reactor main feed water system to illustrate the approach.

3. Proposed work

The presented work is about the analysis of a software system under the fault analysis. The presented work has used the fuzzy logic based system to perform the analysis.

![Figure 1: Module Fault Identification](image)

This work is divided in three main stages. At the very first stage, the complete software system is splitted in smaller modules. Once the modules are identified, the next work is to find the associated faults with each module. Here figure 1 is showing the relationship between software faults and the modules.

Once the faults are defined, the next work is categorize the software faults based on criticality vector. Based on this criticality analysis, some weightage is assigned to each module. There are different criterias to assign the weightage to these faults depending on application type. To assign this weightage, the first work is to identify the module criticality. The modules are divided in three main categories respective to user, system and developer point of view. The module categorization is shown in table 1

<table>
<thead>
<tr>
<th>Table 1: Module Categorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categories</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>User Based</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Module Level</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>System View</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

This kind of prioritization is used to identify most crucial faults in a software system. It means the test cases that represent the crucial faults will be removed first. If no such fault can be removed we cannot continue with the system. The method is quite simple to represent. It is the customize approach in which system will be analyzed respective to the developer point of view. The system will.
be analyzed in terms of some matrices. Once the modules are categorized as well as related faults are identified, the next work is to assign the cost of fault tolerance based on the analysis. Fault assignment based on the module is defined in table 2. The relative assignment is shown here in table 2.

Table 2: Fault Cost Estimation

<table>
<thead>
<tr>
<th>Module Code</th>
<th>Operation/Function</th>
<th>Severity of Defect (S)</th>
<th>Module (M)</th>
<th>SxM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Button Placement</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>GUI design</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Working of Search Button</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Working of Save Button</td>
<td>3</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>Working of Delete</td>
<td>4</td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>6</td>
<td>Backup</td>
<td>3</td>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>Restore</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

Once the fault cost is been identified, the next work is to identify the relationship between the faults. These faults can be aggregative it means the cost of all associated faults will be sum up to find the overall cost of the fault. At the final stage, the fuzzy logic is been implemented to identify the overall cost of the system. The fuzzy logic has estimated the cost under the high, medium and low cost based analysis. The proposed cost estimation algorithm is shown here.

A) Algorithm

1. Define the Software System with N Number of Software Modules
2. Categorize the Modules based on the module type i.e. user, module or system based.
3. For i=1 to N
   { FaultList(i,:) = GenerateFaultList }
4. Assign the Criticality to Software Faults based on Criticality
   For i=1 to Length(Faults)
   { Criticality(i)=Type(Fault(i)) }
5. Estimate the Software Cost based on Fault Analysis per module.
6. For i=1 to N
   { For j=1 to M
     { Cost(i,j)=ModuleRank(i)*Criticality(j) }
   }
7. for i=1 to M
   { For j=1 to M

4. Results

The presented work is implemented in matlab 7.8 environment. The results obtained from the system are given as under

Figure 1: Fault Occurrence in Software Project

Figure 2: Fault Criticality in Software Project
Conclusion

In this present work we have defined a fuzzy based system to estimate the software quality under the criticality vector. Higher the faults in software module as well as higher the fault dependency, more the criticality will be. The obtained results show the clear classification of software modules under criticality vector.

References

Shyi-Ming Chen (2011), Fuzzy group decision making for evaluating the rate of aggregative risk in software development, *Fuzzy Sets and Systems* 118.

Huey-Ming Lee, A new algorithm for applying fuzzy set theory to evaluate the rate of aggregative risk in software development.


Tim Menzies (2000), Practical Large Scale What-if Queries: Case Studies with Software Risk Assessment, 0-7695-0710-7/00© 2000 IEEE


Rattikorn Hewett, Assessment of Software Risks with Model-Based Reasoning.

